7.8 LinkSec

LinkSec (or MACsec, 802.1AE) is a MAC-level encryption/authentication scheme defined in IEEE 802.1AE that uses symmetric cryptography. The 802.1AE defines AES-GCM 128-bit key as a mandatory cipher suite that can be processed by the 82599. The LinkSec implementation, enabled as detailed in Section 4.6.12, supports the following:

- GCM AES 128-bit offload engine in the Tx and Rx data path that support 10 Gb/s wire speed.
- Both host and manageability controller traffic can be processed by the GCM AES engines.
- Support a single, secure Connectivity Association (CA):
  - Single Secure Connection (SC) on transmit data path.
  - Single SC on receive data path.
  - Each SC supports two Security Associations (SAs) for seamless re-keying.
- At any given time, either the manageability controller or the host can act as key agreement entity (KaY – in 802.1AE spec terminology). For example, control and access the offloading engine (SecY in 802.1AE specification terminology).
  - Arbitration semaphores indicate whether the manageability controller or the host acts as the KaY.
  - Tamper resistance — When the manageability controller acts as KaY it can disable accesses from the host to SecY's address space. When the host acts as the KaY no protection is provided.
- Provide statistic counters as listed in the Section 8.3.5.6.
- Support replay protection with replay window equal to zero. Packets that fail replay validation are posted with a replay error in the Rx descriptor. The packets are posted to the host regardless of strict versus check mode described later on in this section.
- Receive memory structure:
  - New LinkSec offload receive status indication in the receive descriptors. LinkSec offload must not be used with the legacy receive format but rather use the extended receive descriptor format.
  - LinkSec header/tag can be posted to the KaY for debug.
- Support VLAN header location according to IEEE 802.1AE (first header inner to the LinkSec tag).
- When LinkSec offload is enabled, Ethernet CRC must be enabled as well by setting both TXCRCEN and RXCRCSTRP bits in the HLREG0 register.

7.8.1 Packet Format

LinkSec defines frame encapsulation format as follows.

Table 7-60 Legacy Frame Format

MAC DA, SA	VLAN (optional)	Legacy Type / Len	LLC data (may include IP/TCP and higher level payload)	CRC
	← - User Data - →			

Table 7-61 LinkSec Encapsulation

MAC DA, SA	LinkSec header (SecTag)	User data (optional encrypted)	LinkSec ICV (tag)	CRC

7.8.2 LinkSec Header (SecTag) Format

Table 7-62 Sectag Format

LinkSec Ethertype	TCI and AN	SL	PN	SCI (optional)
2 bytes	1 byte	1 byte	4 bytes	8 bytes

7.8.2.1 LinkSec Ethertype

The MACsec Ethertype comprises octet 1 and octet 2 of the SecTAG. It is included to allow:

a. Coexistence of MACsec capable systems in the same environment as other systems.

b. Incremental deployment of MACsec capable systems.

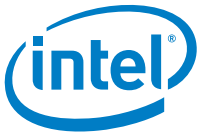
c. Peer SecY's to communicate using the same media as other communicating entities.

d. Concurrent operation of key agreement protocols that are independent of the MACsec protocol and the current cipher suite.

e. Operation of other protocols and entities that make use of the service provided by the SecY's uncontrolled port to communicate independently of the Key agreement state.

Table 7-63 LinkSec Ethertype

Tag Type	Name	Value
802.1AE security TAG	LinkSec Ethertype	88-E5



7.8.2.2 TCI and AN

Table 7-64 TCI and AN Description

Bit(s)	Description
7	Version Number (V). supports only version 0. Packets with other version value are discarded by the 82599.
6	End Station (ES). When set, indicates that the sender is an end station. As a result, SCI is redundant and causes the SC bit to be cleared. Currently, should be always 0b.
5	Secure Channel (SC). Equals 1b when the SCI field is active. If the ES bit is set the SC must be cleared. Since only ES equals zero is supported, the SCI field must be active by setting the LSECTXCTRL.AISCI.
4	Single Copy Broadcast (SCB). Cleared to 0b unless SC supports EPON. Should always be 0b.
3	Encryption (E). Set to 1b when user data is encrypted. (see the note that follows).
2	Changed Text (C). Set to 1b if the data portion is modified by the integrity algorithm. For example, if non-default integrity algorithm is used or if packet is encrypted. (see the note that follows).
1:0	Association Number (AN). 2-bit value defined by control channel to uniquely identify SA (Keys, etc.).

Note: The combination of the *E* bit equals 1b and the *C* bit equals 0b is reserved for KaY packets. The LinkSec logic ignores these packets on the receive path and transfers them to KaY as is (no LinkSec processing and no LinkSec header strip). the 82599's implementation never issues a packet in which the *E* bit is cleared and the *C* bit is set, although can tolerate such packets on receive.

7.8.2.2.1 Short Length

Table 7-65 SL Field Description

Bit(s)	Description
7:6	Reserved, set to 0b.
5:0	Short Length (SL). Number of octets in the secure data field from the end of SecTag to the beginning of ICV if it is less then 48 octets, else SL value is 0b.

7.8.2.2.2 Packet Number (PN)

The LinkSec engine increments it for each packet on the transmit side. The PN is used to generate the initial value (IV) for the crypto engines. When KaY is establishing a new SA, it should set the initial value of PN to 1b. See more details on PN exhausting in Section 7.8.5.1.

7.8.2.3 Secure Channel Identifier (SCI)

The SCI is composed of the Ethernet MAC address and port number as listed in the following table. If the *SC* bit in TCI is not set, the SCI is not encoded in the SecTag.

Table 7-66 SCI Field Description

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Source Ethernet MAC Address						Port Number	

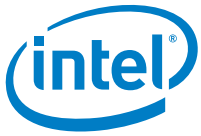
7.8.2.4 Initial Value (IV) Calculation

The IV is the initial value used by the Tx and Rx authentication engines. The IV is generated from the PN and SCI as described in the 802.1AE specification.

7.8.3 LinkSec Management – KaY (Key Agreement Entity)

Kay management is done by the host or the manageability controller. The ownership of LinkSec management is as follows:

1. Initialization at power up or after wake on LAN.
   - In most cases the manageability controller wakes before the host, so:
     - If the manageability controller can be a KaY, it establishes an SC (authentication and key exchange).
     - If the manageability controller cannot be a KaY the only way for it to communicate is through a dedicated Ethernet MAC address or VLAN. This means that the switch must support settings that enable specific Ethernet MAC Address or VLAN to bypass LinkSec.
   - When the host is awake:
     - If the manageability controller acted as KaY, the host should authenticate itself and transfer its ability to authenticate to the manageability controller in order for the manageability controller to transfer ownership over the LinkSec hardware. At this stage, the system operates in proxy mode where the host manages the secured channel while the manageability controller piggybacks on it.



— If the manageability controller wasn't KaY, the host takes ownership over the LinkSec hardware and establishes an SC (authentication and key exchange). The manageability controller mode of operation does not change and it continues to communicate through a dedicated Ethernet MAC address or VLAN.

2. Host in Sx state — manageability controller active:

— If the manageability controller is not Kay capable, then the SC should be reset by link reset or by sending a log-off packet (1af) and then the manageability controller can return to VLAN solution (or remain in such).

— If the manageability controller is KaY capable, the host should notify the manageability controller that it retires KaY ownership and the manageability controller should retake it.

7.8.4 Receive Flow

The 82599 might concurrently receive packets that contain LinkSec encapsulation as well as packets that do not include LinkSec encapsulation. This section describes the incoming packet classification.

- Examine the user data for a SecTAG:
  - — If no SecTag, post the packet with a cleared LinkSec bit in the *Packet Type* field of the receive descriptor.
- Validate frames with a SecTAG:
  - — The MPDU comprises at least 18 octets
  - — Octets 1 and 2 compose the MACsec Ethertype (88E5)
  - — The *V* bit in the TCI is cleared
  - — If the *ES* or the *SCB* bit in the TCI is set, then the *SC* bit is cleared
  - — Bits 7 and 8 of octet 4 of the SecTAG are cleared SL <= 0x3F
  - — If the *C* and *SC* bits in the TCI are cleared, the MPDU comprises 24 octets plus the number of octets indicated by the *SL* field if that is non-zero and at least 72 octets otherwise
  - — If the *C* bit is cleared and the *SC* bit set, then the MPDU comprises 32 octets plus the number of octets indicated by the *SL* field if that is non-zero and at least 80 octets otherwise
  - — If the *C* bit is set and the *SC* bit cleared, then the MPDU comprises 8 octets plus the minimum length of the ICV as determined by the cipher suite in use at the receiving SecY, plus the number of octets indicated by the *SL* field if that is non-zero and at least 48 additional octets otherwise
  - — If the *C* and *SC* bits are both set, the frame comprises at least 16 octets plus the minimum length of the ICV as determined by the cipher suite in use at the receiving SecY, plus the number of octets indicated by the *SL* field if that is non-zero and at least 48 additional octets otherwise
- Extract and decode the SecTAG as specified in Section 7.8.2.
- Extract the user data and ICV as specified section Section 7.8.1.

- Assign the frame to an SA:
  - If a valid SCI, use it to identify the SC
  - Select SA according to AN value
  - If no valid SC or no valid SA found, drop the packet
  - If SCI is omitted, use default SC
  - Select SA according to AN value
  - If no valid SC (or more then SC active) or no valid SA found drop packet
- Perform a preliminary replay check against the last validated PN
- Provide the validation function with:
  - The SA Key (SAK)
  - The SCI for the SC used by the SecY to transmit
  - The PN
  - The SecTAG
  - The sequence of octets that compose the secure data
  - The ICV
- Receive the following parameters from the cipher suite validation operation
  - A valid indication, if the integrity check was valid and the user data could be recovered
  - The sequence of octets that compose the user data
- Update the replay check
- Issue an indication to the controlled port with the DA, SA, and priority of the frame as received from the receive de-multiplexer, and the user data provided by the validation operation

Note: All the references to clauses are to the IEEE P802.1AE/D5.1 document from January 19, 2006.

7.8.4.1 Receive Modes

There are four modes of operation defined for LinkSec Rx as defined by the LSECRXCTRL.LSRXEN field:

1. Bypass (LSRXEN = 00b) — in this mode, LinkSec is not offloaded. There is no authentication or decrypting of the incoming traffic. The LinkSec header and trailer are not removed and these packets are forwarded to the host or the manageability controller according to the regular L2 MAC filtering. The packet is considered as untagged (no VLAN filtering). No further offloads are done on LinkSec packets.
2. Check (LSRXEN = 01b) — in this mode, incoming packets with matching key are decrypted and authenticated according to the LinkSec tag. In this mode both good and erroneous packets are forwarded to host (with the relevant error indication). The only cases where packets are dropped are: erroneous encrypted packets (with the ‘C’ bit in the SecTag header is set) or erroneous packets with replay error if replay protection is enabled in the LSECRXCTRL registers. The Check mode is expected to be

used mainly for debug purposes. In this mode, it may be useful to set also the "Post LinkSec header" bit in the LSECRXCTRL register which controls both SecTag and ICV to be posted to host memory. Note that the header is not removed from KaY packets.

3. Strict (LSRXEN = 10b) — in this mode, incoming packets with matching key are decrypted and authenticated according to the LinkSec tag. The LinkSec header and trailer might be removed from these packets and the packets are forwarded to the host only if the decrypting or authentication was successful. Additional offloads are possible on LinkSec packets. The header is not removed from KaY packets.
4. Drop (LSRXEN = 11b) — in this mode, LinkSec is not offloaded and LinkSec packets are dropped. There is no authentication or decrypting of the incoming traffic.

7.8.4.2 Receive SA Exhausting – Re-Keying

The seamless re-keying mechanism is explained in the following example.

KaY establishes SC0 SC and sets SA0 as the active SA by writing the key in register LinkSec RX Key, writing the AN in LSECRXSA[0], and setting the *SA Valid* bit in the same register. This clears the *Frame Received* bit. On the first packet that arrived to SA0, the frame received automatically sets the *Frame Received* bit. Only at this time the KaY can and should initiate SA1 in the same manner as for SA0. When a frame of SA1 arrives, SA0 retires and can be used for the next SA.

Note: The same mechanism should be used for all RX SCs.

7.8.4.3 Receive SA Context and Identification

Upon arrival of a secured frame the context of the SecTag is verified. This context of the SecTag is described in Section 7.8.2. In order to process the secured frame it should be associated with one of the SA keys. The identification is done by comparing the SCI data with LinkSec RX SC registers and the appropriate SC is selected. To ensure that the SC bit in the TCI of the frame is not set and more than one SC is valid belongs to the frame considered as erroneous and transferred to error handling if only one SC is valid, this SC is selected in this case SC. The incoming frame AN field is compared to the AN field of the Link RX SA register of the selected SC in order to select an SA. The selected SA PN (register LinkSec RX SA PN) field is compared to the incoming PN which should be equal or greater than the LinkSec RX SA PN value, otherwise this frame is dropped. On a match, the selected SA key is used for the secured frame processing.

7.8.4.4 Receive Statistic Counters

A detailed list and description of the LinkSec RX statistics counters can found in Section 8.3.5.6.

7.8.5 Transmit Data Path

The 82599 might concurrently transmit packets that contain LinkSec encapsulation as well as packets that do not include LinkSec encapsulation. This section describes the transmit packet classification, transmit descriptors and statistic counters.

Note: Since flow control (PAUSE) packets are part of the MAC service they should not go through the LinkSec logic.

1. Assign the frame to an SA by adding the AN according to SA select bit in the LSECTXSA register.
2. Assign the next PN variable for that SA to be used as the value of the PN in the SecTAG based on the value in the appropriate (according to SA) LSECTXPN register.
3. Encode the octets of the SecTAG according to the setting in LSECTXCTRL register.
4. Provide the protection function of the current cipher suite with:
   a. The SA Key (SAK).
   b. The SCI for the SC used by the SecY to transmit.
   c. The PN.
   d. The SecTAG.
   e. The sequence of octets that compose the user data.
5. Receive the following parameters from the cipher suite protection operation:
   a. The sequence of octets that compose the secure data.
   b. The ICV.
6. Issue a request to the transmit multiplexer with the destination and source Ethernet MAC addresses, and priority of the frame as received from the controlled port, and an MPDU comprising the octets of the SecTAG, secure data, and the ICV concatenated in that order.

7.8.5.1 Transmit SA Exhausting – Re-Keying

the 82599 supports a single SC on the transmit data path with a seamless re-keying mechanism. The SC might act with one of two optional SAs. The SA is selected statically by the Active SA field in the LSECTXSA register. Once the KaY entity (could be either software or hardware as defined by the LinkSec Ownership field in the LSWFW register) changes the setting of the SA Select field in the LSEXTXSA register the Active SA field is getting the same value on a packet boundary. The next packet that is processed by the transmit LinkSec engine uses the updated SA.

The KaY should switch between the two SAs before the PN is exhausted. In order to protect against such event, hardware generates a LinkSec packet number interrupt to KaY when the PN reaches the exhaustion threshold as defined in the LSECTXCTRL register. The exhaustion threshold should be set to a level that enables the KaY to switch between SA's faster then the PN might be exhausted. If the KaY is slower than it should be, then the PN might be increment above planned. Hardware guarantees that the PN never repeats itself, even if the KaY is slow. Once the PN reaches a value of 0xFF…F0, hardware clears the Enable Tx LinkSec field in the LSECTXCTRL register to 00b. Clearing

the Enable Tx LinkSec field, hardware disables LinkSec offload before the PN could wrap around and then might repeat itself.

Note: Potential race conditions are possible as follows. the 82599 might fetch a transmit packet (indicated as TxPacketN) from the host memory (host or manageability controller packet). KaY can change the setting of the Tx SA Index. The TxPacketN can use the new TX SA Index if the TX SA index was updated before the TxPacketN propagated to the transmit LinkSec engine. This race is not critical since the receiving node should be able to process the previous SA as well as the new SA in the re-keying transition period.

7.8.5.2 Transmit SA Context

Upon transmission of a secured frame, the SA associated data is inserted into the *SecTag* field of the frame. The SecTag data is composed from the LinkSec Tx registers. The SCI value is taken from LinkSec TX SCI Low and High registers unless instructed to omit SCI. The AN value is taken from the active LinkSec TX SA and the PN from the appropriate LinkSec TX SA PN.

7.8.5.3 Transmit Statistic Counters

A detailed list and description of the LinkSec TX statistics counters can found in Section 8.2.3.13.

7.8.6 LinkSec and Manageability

See Section 10.4.

7.8.7 Key and Tamper Protection

LinkSec provides the network administrator protection to the network infrastructure from hostile or unauthorized devices. Since the local host operating system can itself be compromised, hardware protects vital LinkSec context from software access. There are two levels of protection:

- Disable host read access to the LinkSec Keys (keys are write-only)
- Disable host access to LinkSec logic while the firmware manages the LinkSec SC.

7.8.7.1 Key Protection

The LinkSec keys are protected against read accesses at all times. Both software and firmware are not able to read back the keys that hardware uses for transmit and receive activity. Instead, hardware enables the software and firmware reading a signature enabling to verify proper programming of the device. The signature is a byte XOR operation of the Tx and Rx keys readable in the LSECTXSUM and LSECRXSUM fields in the LSECCAP register.

7.8.7.2 Tamper Protection

In a scenario where the host failed authentication and as a result cannot act as the KaY, the manageability controller disables the host access to network and manages the LinkSec channel while the host operating system is already up and running. In such cases, hardware provides the required hooks to protect LinkSec connectivity against hostile software. The manageability controller firmware can disable write accesses generated by the host CPU (on the PCI interface) by setting the *Lock LinkSec Logic* (bit 12) bit in the LSWFW register. Setting this bit can generate an interrupt to the host in case it is enabled by the host in the IMS register.

7.8.8 LinkSec Statistics

7.8.8.1 Rx Statistics

After receiving a packet, one and only one of the statistics in Table 7-67 applies. The precedence order of the statistics is also defined in Table 7-67.

Table 7-67 Rx Statistics

Register Name	802.1ae Name	Priority	Notes
LSECRXBAD	InPktsBadTag	2	Packet is dropped in strict mode or in check mode when the *C* bit is one.
LSECRXUNSCI	InPktsUnknownSCI	3	Used only in check mode. Packet is forwarded to the host if the *C* bit is zero.
LSECRXNOSCI	InPktsNoSCI	3	Packet is dropped in strict mode or in check mode when the *C* bit is one.
LSECRXUNSA	InPktsUnusedSA	4	Packet is dropped in strict mode or in check mode when the *C* bit is one. *Note:* This statistic reflects the sum of InPktsUnusedSA for all SAs.
LSECRXNUSA	InPktsNotUsingSA	4	Used only in check mode. Packet is forwarded to the host if the *C* bit is zero. *Note:* This statistic reflects the sum of InPktsUnusedSA for all SAs.

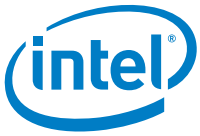


Table 7-67 Rx Statistics (Continued)

Register Name	802.1ae Name	Priority	Notes
LSECRXLATE	InPktsLate	5	
n/a	InPktsOverrun	n/a	The 82599 supports wire-speed decryption and thus this statistic is not needed.
LSECRXNV[SA#]	InPktsNotValid	6	Packet is dropped in strict mode or in check mode when the *C* bit is one.
LSECRXINV[SA#]	InPktsInvalid	6	Used only in check mode. Packet is forwarded to the host if the *C* bit is zero.
LSECRXDELAY	InPktsDelayed	7	
GPRC	InPktsUnchecked	n/a	This statistic is relevant only in bypass mode. In this case, this statistic is reflected in the regular GPRC statistic.
LSECRXOK[SA#]	InPktsOK	8	

7.9 Time SYNC (IEEE1588 and 802.1AS)

7.9.1 Overview

Measurement and control applications are increasingly using distributed system technologies such as network communication, local computing, and distributed objects. Many of these applications are enhanced by having an accurate system-wide sense of time achieved by having local clocks in each sensor, actuator, or other system device. Without a standardized protocol for synchronizing these clocks, it is unlikely that the benefits are realized in the multi-vendor system component market. Existing protocols for clock synchronization are not optimum for these applications. For example, Network Time Protocol (NTP) targets large distributed computing systems with Millisecond (ms) synchronization requirements. The 1588 standard specifically addresses the needs of measurement and control systems:

- Spatially localized
- Microsecond (µs) to sub-µs accuracy
- Administration free
- Accessible for both high-end devices and low-cost, low-end devices

Note: The time sync mechanism activation is possible in full-duplex mode only. There are no limitations on the wire speed although the wire speed might affect the accuracy.

7.9.2 Flow and Hardware/Software Responsibilities

The operation of a Precision Time Protocol (PTP) enabled network is divided into two stages: initialization and time synchronization.

At the initialization stage, every master-enabled node starts by sending sync packets that include the clock parameters of its clock. Upon receipt of a sync packet, a node compares the received clock parameters to its own and if the received parameters are better, then this node moves to a slave state and stops sending sync packets. While in slave state, the node continuously compares the incoming packet to its currently chosen master and if the new clock parameters are better, than the master selection is transferred to this master clock. Eventually the best master clock is chosen. Every node has a defined time-out interval that if no sync packet was received from its chosen master clock it moves back to a master state and starts sending sync packets until a new best master clock (PTP) is chosen.

The time synchronization stage is different to master and slave nodes. If a node is in a master state it should periodically send a sync packet that is time stamped by hardware on the TX path (as close as possible to the PHY). After the sync packet, a Follow_Up packet is sent that includes the value of the time stamp kept from the sync packet. In addition, the master should time stamp Delay_Req packets on its Rx path and return to the slave that sent the time stamp value using a Delay_Response packet. A node in a slave state should time stamp every incoming sync packet and if it came from its

selected master, software uses this value for time offset calculation. In addition, it should periodically send Delay_Req packets in order to calculate the path delay from its master. Every sent Delay_Req packet sent by the slave is time stamped and kept. With the value received from the master with Delay_Response packet, the slave can now calculate the path delay from the master to the slave. The synchronization protocol flow and the offset calculation are shown in Figure 7-36.

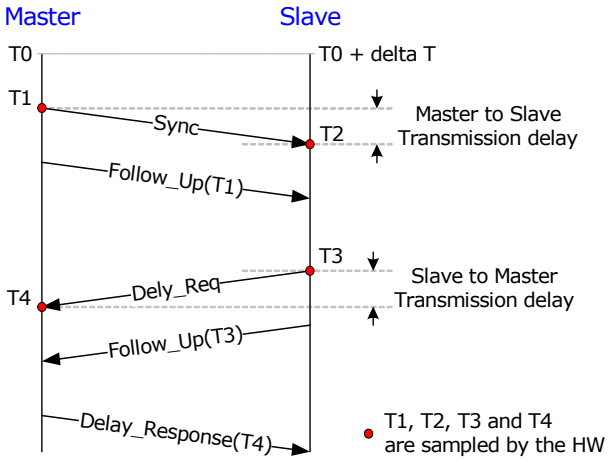


Figure 7-36 Sync Flow and Offset Calculation

Hardware responsibilities are:

1. Identify the packets that require time stamping.
2. Time stamp the packets on both Rx and Tx paths.
3. Store the time stamp value for software.
4. Keep the system time in hardware and give a time adjustment service to software.
5. Maintain auxiliary features related to the system time.

Software responsibilities are:

1. Manageability controller protocol execution, which means defining the node state (master or slave) and selection of the master clock if in slave state.
2. Generate PTP packets, consume PTP packets.
3. Calculate the time offset and adjust the system time using a hardware mechanism for that.
4. Enable configuration and usage of the auxiliary features.

Action	Responsibility	Node Role
Generate a sync packet with time stamp notification in the descriptor.	Software	Master
Time stamp the packet and store the value in registers (T1).	Hardware	Master
Time stamp incoming sync packet, store the value in register and store the sourceID and sequenceID in registers (T2).	Hardware	Slave
Read the time stamp from register put in a Follow_Up packet and send.	Software	Master
Once received, the Follow_Up store T2 from registers and T1 from Follow_up packet.	Software	Slave
Generate a Delay_Req packet with time stamp notification in the descriptor.	Software	Slave
Time stamp the packet and store the value in registers (T3).	Hardware	Slave
Time stamp incoming Delay_Req packet, store the value in register and store the sourceID and sequenceID in registers (T4).	Hardware	Master
Read the time stamp from register and send back to slave using a Delay_Response packet.	Software	Master
Once received, the Delay_Response packet calculate offset using T1, T2, T3 and T4 values.	Software	Slave

7.9.2.1 TimeSync Indications in Rx and Tx Packet Descriptors

Some indications need to be transferred between software and hardware regarding PTP packets. On the Tx path, software should set the 1588 bit in the Tx packet descriptor (bit 9). On the Rx path, hardware has two indications to transfer to software, one is to indicate that this packet is a PTP packet (whether time stamp is taken or not). This is also for other types of PTP packets needed for management of the protocol and this bit is set only for the L2 type of packets (the PTP packet is identified according to its Ethertype). PTP packets have the *L2Type* bit in the packet type field set (bit 9) and the Ethertype matches the filter number set by software to filter PTP packets. The UDP type of PTP packets don't need such indication since the port number (319 for event and 320 all the rest PTP packets) directs the packets toward the time sync application. The second indication is TS (bit 14) to indicate to software that time stamp was taken for this packet. Software needs to access the time stamp registers to get the time stamp values.

7.9.3 Hardware Time Sync Elements

All time sync hardware elements are reset to their initial values (as defined in Section 8.0) upon MAC reset. The clock driving the time sync elements is the DMA clock which frequency depends on the link speed. Upon change in link speed some of the time sync parameters should be changed accordingly. For details please see Table 7-68.

7.9.3.1 System Time Structure and Mode of Operation

The time sync logic contains an up counter to maintain the system time value. This is a 64-bit counter that is built from the SYSTIML and SYSTIMH registers. When operating as a master, the SYSTIMH and SYSTIML registers should be set once by software according to the general system. When operating as a slave, software should update the system time on every sync event as described in Section 7.9.3.3. Setting the system time is done by a direct write to the SYSTIMH register and a fine tune setting of the SYSTIML register using the adjustment mechanism described in Section 7.9.3.3.

Read access to the SYSTIMH and SYSTIML registers should execute in the following manner:

1. Software reads register SYSTIML, at this stage hardware should latch the value of SYSTIMH.
2. Software reads register SYSTIMH, the latched (from last read from SYSTIML) value should be returned by hardware.

Upon an increment event, the system time value should increment its value by the value stored in TIMINCA.*incvalue*. An increment event happens every TIMINCA.*incperiod* cycle if its one then an increment event should occur on every clock cycle. The *incvalue* defines the granularity in which the time is represented by the SYSTMH/L registers. For example, if the cycle time is 16 ns and the *incperiod* is one then and the *incvalue* is 16 then the time is represented in nanoseconds if the *incvalue* is 160 then the time is represented in 0.1 ns units and so on. The *incperiod* helps to avoid inaccuracy in cases where T value cannot be represented as a simple integer and should be multiplied to get to an integer representation. The *incperiod* value should be as small as possible to achieve best accuracy possible.

Table 7-68 Recommended Values for incvalue and incperiod and the outcome SYSTIME

Link Speed	Clock Frequency	Recommended Incvalue	Recommended Incperiod	SYSTIML / SYSTIMH Time Units	SYSTIML / SYSTIMH Granularity
10 Gb/s	156.25 MHz	16000000 (0xF42400)	2	0.8×10^{-15}	12.8 ns
1 Gb/s	15.625 MHz	16000000	2	8×10^{-15}	128 ns
100 Mb/s	1.5625 MHz	16000000	2	80×10^{-15}	1.28 µs

Note: Best accuracy is achieved at lowest permitted Incperiod equals two and as high as possible Incvalue.

7.9.3.2 Time Stamping Mechanism

The time stamping logic is located as close as possible to the PHY. Figure 7-37 shows the exact point in time where the time value is captured by the hardware relative to the packet content. This is to reduce delay uncertainties originated from implementation differences. While the time stamp is sampled at a very late phase in the data path, the 82599 does not insert it to the transferred packet. Instead, the 82599 supports the two-step operation as follows for Tx and Rx.

Tx time stamping

The time stamp logic is activated if enabled by the TSYNCTXCTL.EN bit and the time stamp bit in the packet descriptor is set. In this case, hardware captures the packet's transmission time in the TXSTMPL and TXSTMPH registers. Software is responsible to read the transmission time and append it in the Follow_Up packet as shown in Figure 7-36.

Rx time stamping

On the Rx, this logic parses the traversing frame. If it is matching the message type defined in RXMTRL register, the following packet's parameters are latched: The reception time stamp is stored in the RXSTMPL and RXSTMPH registers. The SourceuuID and SequenceID are stored in the RXSATRL and RXSATRH registers. In addition, two status bits are reported in the Rx descriptor: PTP packet indication (this bit is set only for L2 packets since on the UDP packets the port number direct the packet to the application) and the *TS* bit to identify that a time stamp was taken for this packet (stored in the RXSTMPL and RXSTMPH registers).

Note: The time stamp values are locked in the RXSTMPL and RXSTMPH registers until software accesses them. As long as software does not read these registers, hardware does not capture the time stamp of further Rx packets. In order to avoid potential deadlocks, it is recommended that software read the Rx time stamp registers at some time after sync or Delay_Req packets are expected. It would overcome erroneous cases on which the hardware latches a packet reception time while the packet's content was not posed properly to the software.

Reception consecutive packets that are able to latch its reception time stamp are not supported by the 82599. The RXSATRL and RXSATRH registers may not contain sufficient information to identify uniquely a specific client. Therefore, Master software must not initiate consecutive sync requests before the previous response is received.

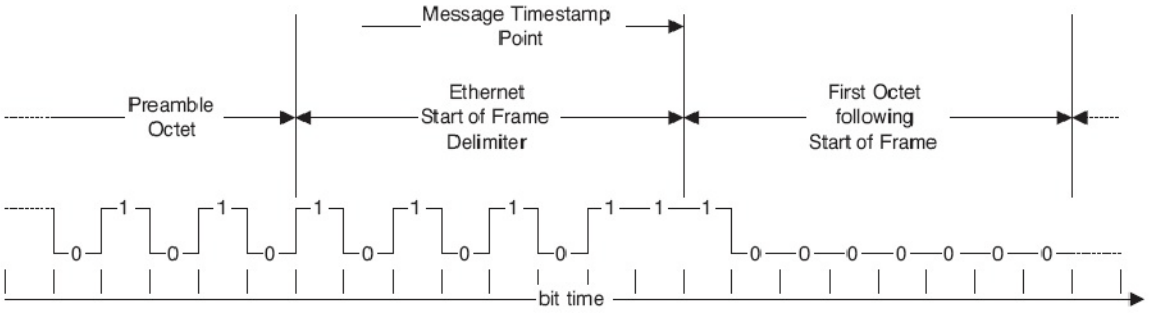
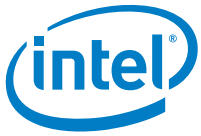


Figure 7-37 Time Stamp Point



7.9.3.3 Time Adjustment Mode of Operation

A node in a time sync network can be in one of two states: master or slave. When a time sync entity is in a master state, it should synchronize other entities to its system clock. In this case, no time adjustments are needed. When the entity is in slave state, it should adjust its system clock by using the data arrived with the Follow_Up and Delay_Response packets and to the time stamp values of Sync and Delay_Req packets. When having all the values software on the slave entity can calculate its offset in the following manner.

After an offset calculation, the system time register should be updated. This is done by writing the calculated offset to TIMADJL and TIMADJH registers. The order should be as follows:

1. Write the lower portion of the offset to TIMADJL.
2. Write the high portion of the offset to TIMADJH to the lower 31 bits and the sign to the most significant bit.

After the write cycle to TIMADJH the value of TIMADJH and TIMADJL should be added to the system time.

7.9.4 Time Sync Related Auxiliary Elements

The time sync logic implements three types of auxiliary element using the precise system timer and SDPs. The time sync block implements two of each features while the possible options of connecting them to SDPs are:

SDP2	Time Stamp 0	Time Stamp 0	Target Time 1	Time Stamp 0	Time Stamp 0	Time Stamp 0
SDP3	Time Stamp 1	Target Time 0	Target Time 0	Time Stamp 1	Time Stamp 1	Target Time 0
SDP6	CLK0	CLK0	CLK0	Target Time 1	Target Time 1	Target Time 1
SDP7	CLK1	CLK1	CLK1	CLK1	Target Time 0	CLK1

Selecting the SDP functionality is done by programming the TimeSync Auxiliary control register and the Extended SDP Control register.

7.9.4.1 Target Time

The target time register is used to get a time triggered event to hardware using an SDP pin. Each target time register is structured the same as the system time register. If the value of the system time is equal to the value written to one of the target time registers, a change in level occurs on one the selected SDP outputs. The accuracy of the comparison is defined by the value of *Mask* field in the TSAUXC register. The target time register also can be used for adjustment of the configurable clock out. Each target time register has an enable bit located in the Auxiliary Control register. After receiving a target time event, the enable bit is cleared and needs to be set again by software to get another target time event.

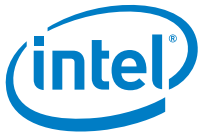
7.9.4.2 Time Stamp Events

After a change in level of an input from one of the SDP pins, a time stamp of the system time is captured into one of the two auxiliary time stamp registers.

7.9.5 PTP Packet Structure

The time sync implementation supports both the 1588 V1 and V2 PTP frame formats. The V1 structure can come only as UDP payload over IPv4 while the V2 can come over L2 with its Ethertype or as a UDP payload over IPv4 or IPv6. The 802.1AS uses only the layer 2 V2 format. Note that PTP frame structure over UDP is not supported in the 82599 for IP tunneling packets.

<table>
<tr><th>Offset in Bytes</th><th>V1 Fields</th><th colspan="2">V2 Fields</th></tr>
<tr><td>Bits</td><td>7 6 5 4 3 2 1 0</td><td colspan="2">7 6 5 4 3 2 1 0</td></tr>
<tr><td>0</td><td rowspan="2">versionPTP</td><td>transportSpecific[1]</td><td>messageId</td></tr>
<tr><td>1</td><td>Reserved</td><td>versionPTP</td></tr>
<tr><td>2</td><td rowspan="2">versionNetwork</td><td colspan="2" rowspan="2">messageLength</td></tr>
<tr><td>3</td></tr>
<tr><td>4</td><td rowspan="16">Subdomain</td><td colspan="2">SubdomainNumber</td></tr>
<tr><td>5</td><td colspan="2">Reserved</td></tr>
<tr><td>6</td><td colspan="2" rowspan="2">flags</td></tr>
<tr><td>7</td></tr>
<tr><td>8</td><td colspan="2" rowspan="8">Correction Field</td></tr>
<tr><td>9</td></tr>
<tr><td>10</td></tr>
<tr><td>11</td></tr>
<tr><td>12</td></tr>
<tr><td>13</td></tr>
<tr><td>14</td></tr>
<tr><td>15</td></tr>
<tr><td>16</td><td colspan="2" rowspan="4">Reserved</td></tr>
<tr><td>17</td></tr>
<tr><td>18</td></tr>
<tr><td>19</td></tr>
</table>



Offset in Bytes	V1 Fields	V2 Fields
Bits	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
20	messageType	Source Port ID (only part of the field is captured in the RXSATRL and RXSATRH registers)
21	Source communication technology	
22	Sourceuuid	
23		
24		
25		
26		
27		
28	sourceportid	
29		
30	sequenceId	sequenceId
31		
32	control	control
33	reserved	logMessagePeriod
34	falgs	n/a
35		

1. Should all be zero.

Note: Only the fields with the bold italic format colored red are of interest to hardware.

Ethernet (L2)	VLAN (Optional)	PTP Ethertype	PTP message
Ethernet (L2)	IP (L3)	UDP	PTP message

When a PTP packet is recognized (by Ethertype or UDP port address) on the Rx side the version should be checked if it is V1 then the control field at offset 32 should be compared to message field in register described in Section 8.2.3.26.6, otherwise the byte at offset 0 should be used for comparison to the rest of the needed field are at the same location and size for both V1 and V2.

Enumeration	Value
PTP_SYNC_MESSAGE	0
PTP_DELAY_REQ_MESSAGE	1
PTP_FOLLOWUP_MESSAGE	2
PTP_DELAY_RESP_MESSAGE	3
PTP_MANAGEMENT_MESSAGE	4
reserved	5–255

MessageId	Message Type	Value (hex)
PTP_SYNC_MESSAGE	Event	0
PTP_DELAY_REQ_MESSAGE	Event	1
PTP_PATH_DELAY_REQ_MESSAGE	Event	2
PTP_PATH_DELAY_RESP_MESSAGE	Event	3
Unused		4-7
PTP_FOLLOWUP_MESSAGE	General	8
PTP_DELAY_RESP_MESSAGE	General	9
PTP_PATH_DELAY_FOLLOWUP_MESSAGE	General	A
PTP_ANNOUNCE_MESSAGE	General	B
PTP_SIGNALLING_MESSAGE	General	C
PTP_MANAGEMENT_MESSAGE	General	D
Unused		E-F

If V2 mode is configured in Section 8.2.3.26.15,then time stamp should be taken on PTP_PATH_DELAY_REQ_MESSAGE and PTP_PATH_DELAY_RESP_MESSAGE for any value in the message field in the register described at Section 8.2.3.26.6.

7.10 Virtualization

7.10.1 Overview

I/O virtualization is a mechanism that can be used to share I/O resources among several consumers. For example, in a virtual system, multiple operating systems are loaded and each operates as though the entire system's resources were at its disposal. However, for the limited number of I/O devices, this presents a problem because each operating system might be in a separate memory domain and all the data movement and device management has to be done by a Virtual Machine Monitor (VMM). VMM access adds latency and delay to I/O accesses and degrades I/O performance. Virtualized devices are designed to reduce the burden of the VMM by making certain functions of an I/O device shared among multiple guest operating systems or a Virtual Machine (VM), thereby allowing each VM direct access to the I/O device.

The 82599 supports two modes of operations of virtualized environments:

1. Direct assignment of part of the port resources to different guest operating systems using the PCI SIG SR IOV standard. Also known as native mode or pass through mode. This mode is referenced as IOV mode throughout this section.
2. Central management of the networking resources by an IOVM or by the VMM. Also known as software switch acceleration mode. This mode is referred to as Next Generation VMDq mode in this section.

The virtualization offloads capabilities provided by the 82599 apart from the replication of functions defined in the PCI SIG IOV specification are part of Next Generation VMDq.

A hybrid model, where part of the VMs are assigned a dedicated share of the port and the rest are serviced by an IOVM is also supported. However, in this case the offloads provided to the software switch might be more limited. This model can be used when parts of the VMs run operating systems for which VF drivers are available and thus can benefit from an IOV and others that run older operating systems for which VF drivers are not available and are serviced by an IOVM. In this case, the IOVM is assigned one VF and receives all the packets with Ethernet MAC addresses of the VMs behind it.

The following section describes the support the 82599 provides for these modes.

This section assumes a single-root implementation of IOV and no support for multi-root.

7.10.1.1 Direct Assignment Model

The direct assignment support in the 82599 is built according to the following model of the software environment.

It is assumed that one of the software drivers sharing the port hardware behaves as a master driver (Physical Function or PF driver). This driver is responsible for the initialization and the handling of the common resources of the port. All the other drivers (Virtual Function drivers or VF drivers) might read part of the status of the common parts but cannot change them. The PF driver might run either in the VMM or in some service operating system. It might be part of an IOVM or part of a dedicated service operating system.

In addition, part of the non time-critical tasks are also handled by the PF driver. For example, access to CSR through the I/O space or access to the configuration space are available only through the master interface. Time-critical CSR space like control of the Tx and Rx queue or interrupt handling is replicated per VF, and directly accessible by the VF driver.

Note: In some systems with a thick hypervisor, the service operating system might be an integral part of the VMM. For these systems, each reference to the service operating system in the sections that follow refer to the VMM.

7.10.1.1.1 Rationale

The direct assignment model enables each of the VMs to receive and transmit packets with minimum of overhead. Non time-critical operations such as initialization and error handling can be done via the PF driver. In addition, it is important that the VMs can operate independently with minimal disturbance. It is also preferable that the VM interface to hardware should be as close as possible to the native interface in non-virtualized systems in order to minimize the software development effort.

The main time critical operations that require direct handling by the VM are:

- Maintenance of the data buffers and descriptor rings in host memory. In order to support this, the DMA accesses of the queues associated to a VM should be identified as such on the PCIe using a different requester ID.
- Handling of the hardware ring (tail bump and head updates)
- Interrupts handling

The capabilities needed to provide independence between VMs are:

- Per VM reset and enable capabilities
- Tx rate control
- Allocating separate CSR space per VM. This CSR space is organized as close as possible to the regular CSR space to enable sharing of the base driver code.

Note: The rate control and VF enable capabilities are controlled by the PF.

7.10.1.2 System Overview

The following drawings show the various elements involved in the I/O process in a virtualized system. Figure 7-38 shows the flow in software Next Generation **VMDq** mode and Figure 7-39 shows the flow in IOV mode.

This section assumes that in IOV mode, the driver on the guest operating system is aware that it operates in a virtual system (para-virtualized) and there is a channel between each of the VM drivers and the PF driver allowing message passing such as configuration request or interrupt messages. This channel can use the mailbox system implemented in the 82599 or any other means provided by the VMM vendor.

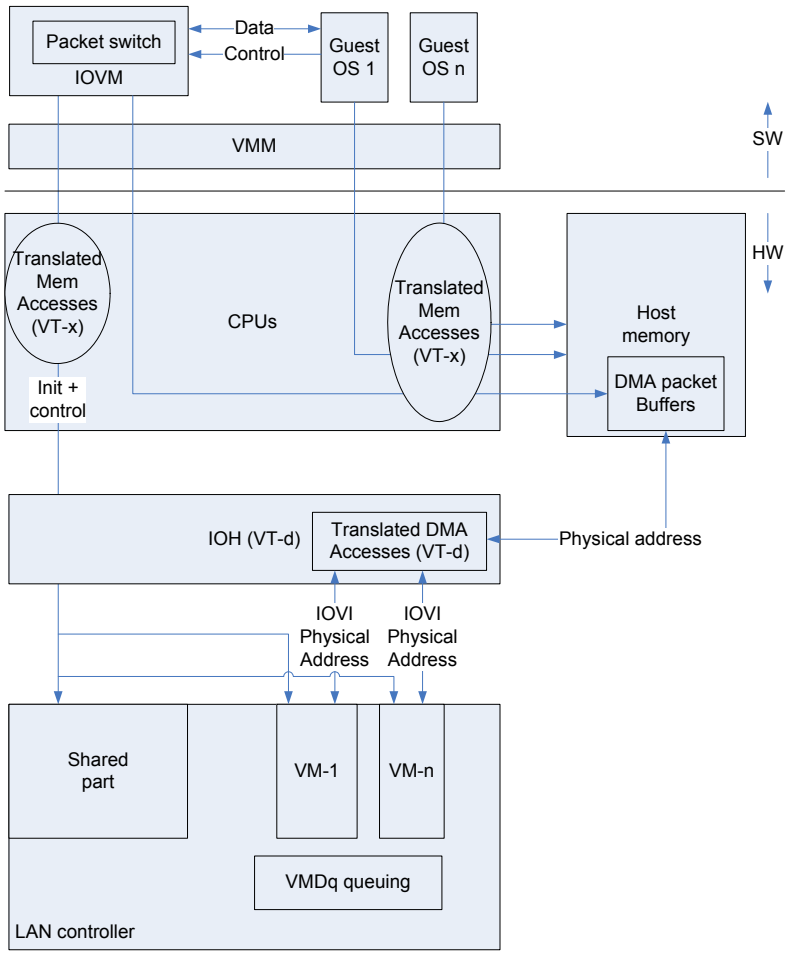


Figure 7-38 System Configuration for Next Generation VMDq Mode

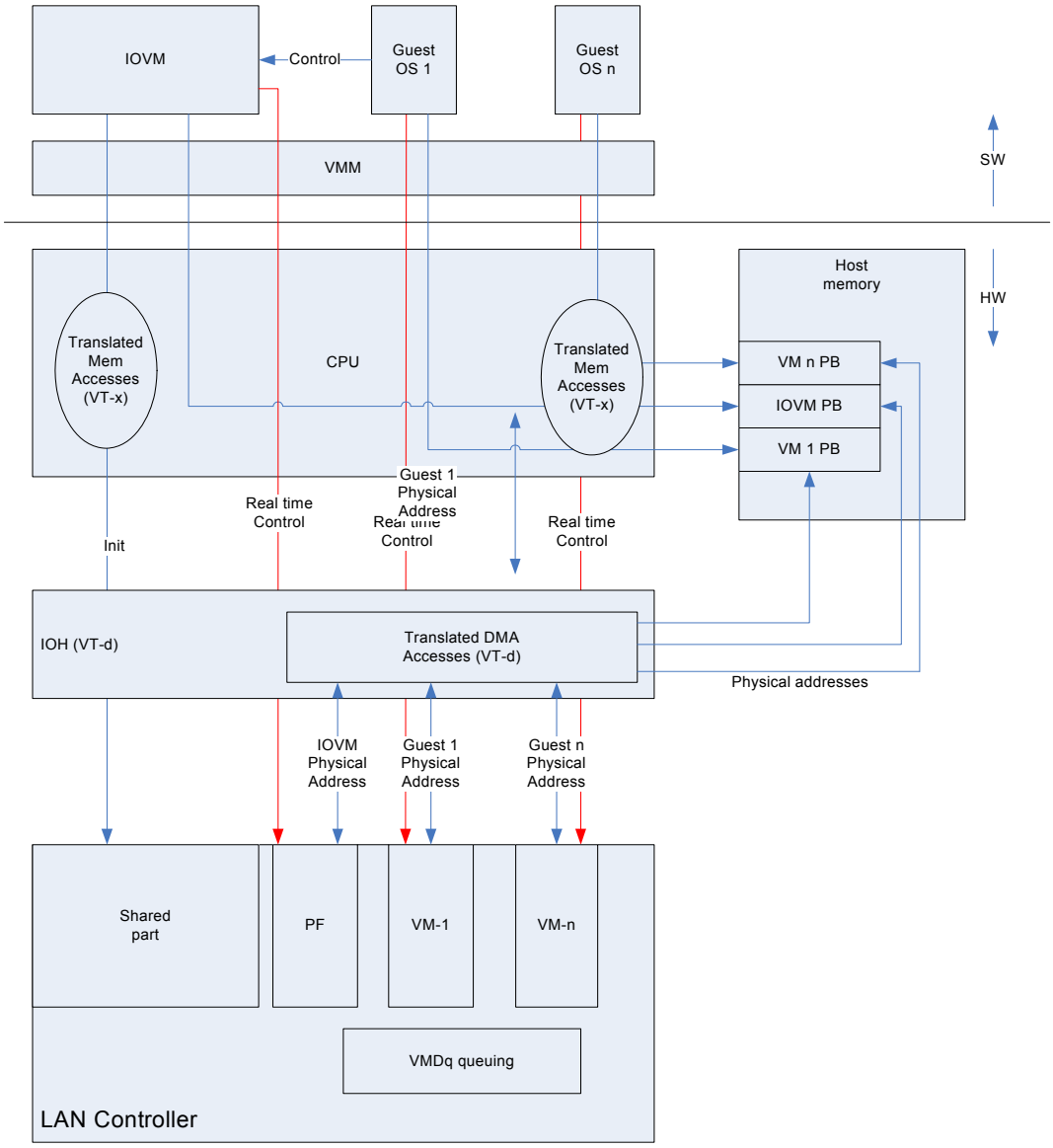
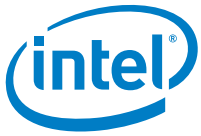


Figure 7-39 System Configuration for IOV Mode



7.10.2 PCI-SIG SR-IOV Support

7.10.2.1 SR-IOV Concepts

SR-IOV defines the following entities in relation to I/O virtualization:

- Virtual Image (VI): Part of the I/O resources are assigned to a A VM.
- I/O Virtual Intermediary (IOVI) or I/O Virtual Machine (IOVM): A special VM that owns the physical device and is responsible for the configuration of the physical device.
- Physical function (PF): A function representing a physical instance — One port for the 82599. The PF driver is responsible for the configuration and management of the shared resources in the function.
- Virtual Function (VF): A part of a PF assigned to a VI.

7.10.2.2 Configuration Space Replication

The SR-IOV specification defines a reduced configuration space for the virtual functions. Most of the PCIe configuration of the VFs comes from the PF.

This section describes the expected handling of the different parts of the configuration space for virtual functions. It deals only with the parts relevant to the 82599.

Details of the configuration space for virtual functions can be found in Section 9.5.

7.10.2.2.1 Legacy PCI Configuration Space

The legacy configuration space is allocated to the PF only and emulated for the VFs. A separate set of BARs and one bus master enable bit is allocated in the SR-IOV capability structure in the PF and is used to define the address space used by the entire set of VFs.

All the legacy error reporting bits are emulated for the VF. See Section 7.10.2.4 for details.

7.10.2.2.2 Memory BARs Assignment

The SR-IOV specification defines a fixed stride for all the VF BARs, so that each VF can be allocated part of the memory BARs at a fixed stride from the a basic set of BARs. In this method, only two decoders per replicated BAR per PF are required and the BARs reflected to the VF are emulated by the VMM.

The only BARs that are useful for the VFs are BAR0 and BAR3, so only those are replicated. The following table lists the existing BARs and the stride used for the VFs:

Table 7-69 BARs in the 82599 (64-bit BARs)

BAR	Type	Usage	Requested Size per VF (=Stride)
0, 1	Mem	CSR space	Maximum (16 KB, page size). For page size see Section 9.4.4.8 for more details.
2	n/a	Not used	n/a
3, 4	Mem	MSI-X	Maximum (16 KB, page size).
5	n/a	Not used	n/a

BAR0 of the VFs are a sparse version of the original PF BAR and include only the register relevant to the VF. For more details see Section 7.10.2.7.

Figure 7-40 shows the different BARs in an IOV-enabled system:

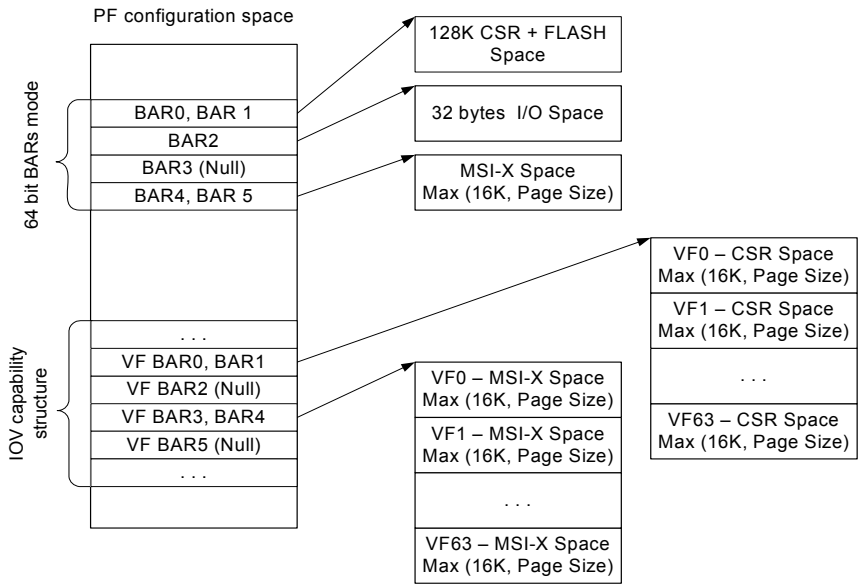


Figure 7-40 BARs in an IOV-enabled System

7.10.2.2.3 PCIe Capability Structure

The PCIe capability structure is shared between the PF and the VFs. The only relevant bits that are replicated are:

1. Transaction pending
2. Function Level Reset (FLR). See Section 7.10.2.3 for details.