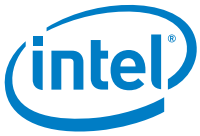


Field	Bit(s)	Init Val	Description
Reserved	15	0b	Reserved.
WE	16	0b	Write Enable. When this bit is set, the content of bits 3:0 are written into the relevant queue context. Bits 3:1 are reserved. This bit should never be set together with the *RE* bit in this register.
RE	17	0b	Read Enable. When this bit is set, the content of bits 3:0 are read from the relevant queue context. Bits 3:1 are reserved. This bit should never be set together with the *WE* bit in this register.
Reserved	31:18	0x0	Reserved.

8.2.3.27.10 PF VM Tx Switch Loopback Enable — PFVMTXSW[n] (0x05180 + 4*n, n=0...1; RW)

Field	Bit(s)	Init Val	Description
LLE	31:0	0x0	Local Loopback Enable. For each register 'n', and bit 'i', i=0..31, enables Local loopback for pool 32*n+1. When set, a packet originating from a specific pool and destined to the same pool is allowed to be looped back. If cleared, the packet is dropped.

8.2.3.27.11 PF VF Anti Spoof Control — PFVFSPOOF[n] (0x08200 + 4*n, n=0...7; RW)

Field	Bit(s)	Init Val	Description
MACAS	7:0	0x0	For each register 'n', and bit 'i', i=0..7, enables anti-spoofing filter on Ethernet MAC addresses for VF(8*n+1).
VLANAS	15:8	0x0	For each register 'n', and bit '8+i', i=0..7, enables anti-spoofing filter on VLAN tag for VF(8*n+i). *Note:* If *VLANAS* is set for a specific pool, then the respective *MACAS* bit must be set as well.
Reserved	31:16	0x0	Reserved.

8.2.3.27.12 PFDMA Tx General Switch Control — PFDTXGSWC (0x08220; RW)

Field	Bit(s)	Init Val	Description
LBE	0	0b	Enables VMDQ loopback.
Reserved	31:1	0x0	Reserved.

8.2.3.27.13 PF VM VLAN Insert Register — PFVMVIR[n] (0x08000 + 4*n, n=0...63; RW)

Field	Bit(s)	Init Val	Description
Port VLAN ID	15:0	0x0	Port VLAN tag to insert if the *VLANA* field = 01b.
Reserved	29:16	0x0	Reserved.
VLANA	31:30	0x0	VLAN action. 00b = Use descriptor command. 01b = Always insert default VLAN. 10b = Never insert VLAN. 11b = Reserved.

8.2.3.27.14 PF VM L2 Control Register — PFVML2FLT[n] (0x0F000 + 4*n, n=0...63; RW)

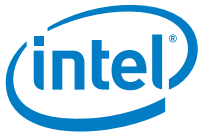
This register controls per VM Inexact L2 Filtering.

Field	Bit(s)	Init Val	Description
Reserved	23:0	0x0	Reserved.
AUPE	24	0b	Accept Untagged Packets Enable. When set, packets without a VLAN tag can be forwarded to this queue, assuming they pass the Ethernet MAC address queuing mechanism.
ROMPE	25	0b	Receive Overflow Multicast Packets. Accept packets that match the MTA table.
ROPE	26	0b	Receive MAC Filters Overflow. Accept packets that match the PFUTA table.
BAM	27	0b	Broadcast Accept.
MPE	28	0b	Multicast Promiscuous.
Reserved	31:29	0x0	Reserved.

8.2.3.27.15 PF VM VLAN Pool Filter — PFVLVF[n] (0x0F100 + 4*n, n=0...63; RW)

Software should initialize these registers before transmit and receive are enabled.

Field	Bit(s)	Init Val	Description
VLAN_Id	11:0	X	Defines a VLAN tag for pool VLAN filter n. The bitmap defines which pools belong to this VLAN. *Note:* Appears in little endian order (LS byte last on the wire).



Field	Bit(s)	Init Val	Description
Reserved	30:12	X	Reserved.
VI_En	31	X	VLAN Id Enable — this filter is valid.

8.2.3.27.16 PF VM VLAN Pool Filter Bitmap — PFVLVFB[n] (0x0F200 + 4*n, n=0...127; RW)

Software should initialize these registers before transmit and receive are enabled.

Field	Bit(s)	Init Val	Description
POOL_ENA	31:0	x	Pool Enable Bit Array. Each couple of registers '2*n' and '2*n+1' enables routing of packets that match a PFVLVF[n] filter to a pool list. Each bit when set, enables packet reception with the associated pools as follows: • Bit 'i' in register '2*n' is associated with POOL 'i'. • Bit 'i' in register '2*n+1' is associated with POOL '32+i'.

8.2.3.27.17 PF Unicast Table Array — PFUTA[n] (0x0F400 + 4*n, n=0...127; RW)

There is one register per 32 bits of the unicast address table for a total of 128 registers (the PFUTA[127:0] designation). Software must mask to the desired bit on reads and supply a 32-bit word on writes. The first bit of the address used to access the table is set according to the MCSTCTRL.MO field.

The seven MS bits of the Ethernet MAC address (out of the 12 bits) selects the register index while the five LS bits (out of the 12 bits) selects the bit within a register.

Note: All accesses to this table must be 32 bit.

The look-up algorithm is the same one used for the MTA table.

This table should be zeroed by software before start of operation.

Field	Bit(s)	Init Val	Description
Bit Vector	31:0	X	Word wide bit vector specifying 32 bits in the unicast destination address filter table.

8.2.3.27.18 PF Mirror Rule Control — PFMRCTL[n] (0x0F600 + 4*n, n= 0...3; RW)

This register defines mirroring rules for each of four destination pools.

Field	Bit(s)	Init Val	Description
VPME	0	0b	Virtual Pool Mirroring Enable. Enables mirroring of certain pools as defined in the PFMRVM registers.
UPME	1	0b	Uplink Port Mirroring Enable. Enables mirroring of all traffic received from the network.
DPME	2	0b	Downlink Port Mirroring Enable. Enables mirroring of all traffic transmitted to the network.
VLME	3	0b	VLAN Mirroring Enable. Enables mirroring of a set of given VLANs as defined in the PFMRVLAN registers.
Reserved	7:4	0x0	Reserved.
MP	13:8	0x0	Mirror Pool. Defines the destination pool for this mirror rule.
Reserved	31:14	0x0	Reserved.

8.2.3.27.19 PF Mirror Rule VLAN — PFMRVLAN[n] (0x0F610 + 4*n, n= 0...7; RW)

This register defines the VLAN values as listed in the PFVLVF table taking part in the VLAN mirror rule.

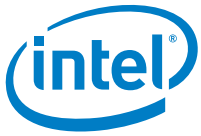
Registers 0, 4 correspond to rule 0, registers 1, 5 correspond to rule 1, etc. Registers 0-3 correspond to the LSB in the PFVLVF table. For example, register 0 corresponds to VLAN filters 31:0, while register 4 corresponds to VLAN filters 63:32.

Field	Bit(s)	Init Val	Description
VLAN	31:0	0x0	Bitmap listing which VLANs participate in the mirror rule.

8.2.3.27.20 PF Mirror Rule Pool — PFMRVM[n] (0x0F630 + 4*n, n= 0...7; RW)

This register defines which pools are being mirrored to the destination pool.

Registers 0, 4 correspond to rule 0, registers 1, 5 correspond to rule 1, etc. Registers 0-3 correspond to the LSB in the pool list. For example, register 0 corresponds to pools 31:0, while register 4 corresponds to pools 63:32.



Field	Bit(s)	Init Val	Description
Pool	31:0	0x0	Bitmap listing which pools participate in the mirror rule.

8.3 Device Registers — VF

8.3.1 Registers Allocated Per Queue

Depending on configuration, each pool has 2, 4, or 8 queues allocated to it. Note that in IOV mode, any queues not allocated to a VF are allocated to the PF. The registers assigned to a queue are accessible both in its VF address space and in the PF address space. This section describes the address mapping of registers that belong to queues.

Section 7.10.2.7.2 defines the correspondence of queue indices between the PF and the VFs. For example, when in configuration for 32 VFs, queues 124-127 in the PF correspond to queues [3:0] of VF# 31.

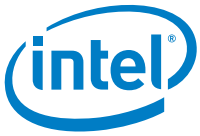
The queues are enumerated in each VF from 0 (such as [1:0], [3:0], or [7:0]). If a queue is allocated to a VF, then its corresponding registers are accessible in the VF CSR space. Each register is allocated an address in the VF (relative to its base) according to its index in the VF space. Therefore, the registers of queue 0 in each VF are allocated the same addresses, which equal the addresses of the same registers for queue 0 in the PF. For example, RDH[0] in the VF space has the same relative address in each VF and in the PF (address 0x01010).

8.3.2 Non-Queue Registers

Registers that do not correspond to a specific queue are allocated addresses in the VF space according to these rules:

- Registers that are read only by the VF (like STATUS) have the same address in the VF space as in the PF space.
- Registers allocated per pool are accessed in the VF in the same location as pool [0] in the PF address space.
- Registers that are read/write by the VF (like CTRL) are replicated in the PF, one per VF, in adjacent addresses.

Note: Since the VF address space is limited to 16 KB, any register that resides above that address in the PF space cannot reside in the same address in the VF space and is therefore allocated in another location in the VF.



8.3.3 MSI—X Register Summary VF — BAR 3

Virtual Address	Physical Address Base (+ VFn *0x30)	Abbreviation	Name
0x0000 + n*0x10, n=0...2	0x00010	MSIXTADD	MSIX Table Entry Lower Address
0x0004 + n*0x10, n=0...2	0x00018	MSIXTUADD	MSIX Table Entry upper Address
0x0008 + n*0x10, n=0...2	0x00028	MSIXTMSG	MSIX Table Entry Message
0x000C + n*0x10, n=0...2	N/A	MSIXTVCTRL	MSIX Table Vector Control
Max(Page Size, 0x2000)	N/A	MSIXPBA	MSI-X Pending Bit Array

8.3.3.1 MSI—X Table Entry Lower Address — MSIXTADD (BAR3: 0x0000 + n*0x10, n=0...2; RW)

See Section 9.3.8.2 for details of this register.

8.3.3.2 MSI—X Table Entry Upper Address — MSIXTUADD (BAR3: 0x0004 + n*0x10, n=0...2; RW)

See Section 9.3.8.2 for details of this register.

8.3.3.3 MSI—X Table Entry Message — MSIXTMSG (BAR3: 0x0008 + n*0x10, n=0...2; RW)

See Section 9.3.8.2 for details of this register.

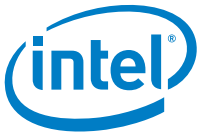
8.3.3.4 MSI—X Table Entry Vector Control — MSIXVCTRL (BAR3: 0x000C + n*0x10, n=0...2; RW)

See Section 9.3.8.2 for details of this register.

8.3.3.5 MSIXPBA (BAR3: 0x02000; RO) — MSIXPBA Bit Description

Field	Bit(s)	Init Val	Description
Pending Bits	2:0	0x0	For each pending bit that is set, the function has a pending message for the associated MSI-X table entry. Pending bits that have no associated MSI-X table entry are reserved.
Reserved	31:3	0x0	Reserved

Note: If a page size larger than 8 KB is programmed in the IOV structure, the address of the MSIX PBA table moves to be page aligned.

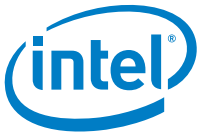


8.3.4 Registers Summary VF — BAR 0

8.3.4.1 VF Registers Table

Virtual Address	Abbreviation	Name	Block	Reset Source	RW
General Control Registers					
0x00000	VFCTRL	VF Control Register	Target		WO
0x00008	VFSTATUS	VF Status Register	Target		RO
0x00010	VFLINKS	VF Link Status Register	MAC		RO
0x00048	VFFRTIMER	VF Free Running Timer	Rx-Filter		RO
0x002FC	VFMailbox	VF Mailbox	Target		RW
0x00200 + 4*n, n=0...15	VFMBMEM[n]	VF Mailbox Memory	Target		RW
0x03190	VFRXMEMWRAP	VF Rx Packet Buffer Flush Detect	DBU-Rx		RO
Interrupt Registers					
0x00100	VFEICR	VF Extended Interrupt Cause	Interrupt		RC/W1C
0x00104	VFEICS	VF Extended Interrupt Cause Set	Interrupt		WO
0x00108	VFEIMS	VF Extended Interrupt Mask Set/Read	Interrupt		RWS
0x0010C	VFEIMC	VF Extended Interrupt Mask Clear	Interrupt		WO
0x00110	VFEIAC	VF Extended Interrupt Auto Clear	Interrupt		RW
0x00114	VFEIAM	VF Extended Interrupt Auto Mask Enable	Interrupt		RW
0x00820 + 4*n, n=0...1	VFEITR	VF Extended Interrupt Mask Set/Read	Interrupt		RWS
0x00120 + 4*n, n=0...3	VFIVAR	VF Interrupt Vector Allocation Registers	Interrupt		RW
0x00140	VFIVAR_MISC	VF Interrupt Vector Allocation Registers	Interrupt		RW
0x00180 + 4*n, n=0,1	VFRSCINT	VF RSC Enable Interrupt	Interrupt		RW
0x00148	VFPBACL	VF MSI—X PBA Clear	PCIe		RW1C
Receive DMA Registers					
0x01000 + 0x40*n, n=0...7	VFRDBAL	VF Receive Descriptor Base Address Low	DMA-Rx		RW

Virtual Address	Abbreviation	Name	Block	Reset Source	RW
0x01004 + 0x40*n, n=0...7	VFRDBAH	VF Receive Descriptor Base Address High	DMA-Rx		RW
0x01008 + 0x40*n, n=0...7	VFRDLEN	VF Receive Descriptor Ring Length	DMA-Rx		RW
0x01010 + 0x40*n, n=0...7	VFRDH	VF Receive Descriptor Head	DMA-Rx		RO
0x01018 + 0x40*n, n=0...7	VFRDT	VF Receive Descriptor Tail	DMA-Rx		RW
0x01028 + 0x40*n, n=0...7	VFRXDCTL	VF Receive Descriptor Control	DMA-Rx		RW
0x01014 + 0x40*n, n=0...7	VFSRRCTL	VF Split and Replication Receive Control Register queue	DMA-Rx		RW
0x00300	VFPSRTYPE	VF Replication Packet Split Receive Type	DBU-Rx		RW
0x0102C + 0x40*n, n=0...7	VFRSCCTL	VF RSC Control	DMA-Rx		RW
Transmit DMA Registers					
0x02000 + 0x40*n, n=0...7	VFTDBAL	VF Transmit Descriptor Base Address Low	DMA-Tx		RW
0x02004 + 0x40*n, n=0...7	VFTDBAH	VF Transmit Descriptor Base Address High	DMA-Tx		RW
0x02008 + 0x40*n, n=0...7	VFTDLEN	VF Transmit Descriptor Ring Length	DMA-Tx		RW
0x02010 + 0x40*n, n=0...7	VFTDH	VF Transmit Descriptor Head	DMA-Tx		RO
0x02018 + 0x40*n, n=0...7	VFTDT	VF Transmit Descriptor Tail	DMA-Tx		RW
0x02028 + 0x40*n, n=0...7	VFTXDCTL	VF Transmit Descriptor Control	DMA-Tx		RW
0x02038 + 0x40*n, n=0...7	VFTDWBAL	VF Tx Descriptor Completion Write-Back Address Low	DMA-Tx		RW
0x0203C + 0x40*n, n=0...7	VFTDWBAH	VF Tx Descriptor Completion Write-Back Address High	DMA-Tx		RW
DCA Registers					
0x0100C + 0x40*n, n=0...7	VFDCA_RXCTRL	VF Rx DCA Control Registers	DMA-Rx		RW
0x0200C + 0x40*n, n=0...7	VFDCA_TXCTRL	VF Tx DCA Control Registers	DMA-Tx		RW
Statistic Register					
0x0101C	VFGPRC	VF Good Packets Received Count	DMA-Rx		RO
0x0201C	VFGPTC	VF Good Packets Transmitted Count	STAT		RO
0x01020	VFGORC_LSB	VF Good Octets Received Count Low	DMA-Rx		RO
0x01024	VFGORC_MSB	VF Good Octets Received Count High	DMA-Rx		RO



Virtual Address	Abbreviation	Name	Block	Reset Source	RW
0x02020	VFGOTC_LSB	VF Good Octets Transmitted Count Low	STAT		RO
0x02024	VFGOTC_MSB	VF Good Octets Transmitted Count High	STAT		RO
0x01034	VFMPRC	VF Multicast Packets Received Count	DMA-Rx		RO

8.3.5 Detailed Register Descriptions —VF

All the registers in this section are replicated per VF. The addresses are relative to the beginning of each VF address space. The address relative to BAR0 as programmed in the IOV structure in the PF configuration space (offset 0x180-0x184) can be found by the following formula:

```
VF BAR0 + Max(16K, system page size)* VF# + CSR offset.
```

8.3.5.1 General Control Registers —VF

8.3.5.1.1 VF Control Register — VFCTRL (0x00000; WO)

Field	Bit(s)	Init Val	Description
Reserved	25:0	0x0	Reserved.
RST	26	0b	VF Reset. This bit performs a reset of the queue enable and the interrupt registers of the VF.
Reserved	31:27	0x0	Reserved

8.3.5.1.2 VF Status Register — VFSTATUS (0x00008; R)

This register is a mirror of the PF status register. See Section 8.2.3.1.2 for details of this register.

8.3.5.1.3 VF Link Status Register — VFLINKS (0x00010; RO)

This register is a mirror of the PF LINKS register. See Section 8.2.3.22.20 for details of this register.

8.3.5.1.4 VF Free Running Timer — VFFRTIMER (0x00048; RO)

This register mirrors the value of a free running timer register in the PF — RTFRTIMER. The register is reset by a PCI reset and/or software reset. This register is a mirror of the PF register.

8.3.5.1.5 VF Mailbox — VFMailbox (0x002FC; RW)

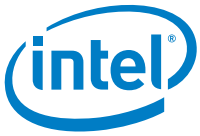
Field	Bit(s)	Init Val	Description
Req (WO)	0	0b	Request for PF ready. Setting this bit, causes an interrupt to the PF. This bit always reads as zero. Setting this bit sets the corresponding bit in *VFREQ* field in PFMBICR register.
Ack (WO)	1	0b	PF message received. Setting this bit, causes an interrupt to the PF. This bit always reads as zero. Setting this bit sets the corresponding bit in *VFACK* field in PFMBICR register.
VFU	2	0b	Buffer is taken by VF. This bit can be set only if the *PFU* bit is cleared and is mirrored in the *VFU* bit of the PFMailbox register.
PFU	3	0b	Buffer is taken by PF. This bit is RO for the VF and is a mirror of the *PFU* bit of the PFMailbox register.
PFSTS (RC)	4	0b	PF wrote a message in the mailbox.
PFACK (RC)	5	0b	PF acknowledged the VF previous message.
RSTI (RO)	6	1b	Indicates that the PF had reset the shared resources and the reset sequence is in progress.
RSTD (RC)	7	0b	Indicates that a PF software reset completed. This bit is cleared on read.
Reserved	31:8	0x0	Reserved.

Note: *VFLR* won't clear the VFMAILBOX.VFU bit. This bit should be cleared by a direct write access or by setting PFMailbox.RVFU bit.

8.3.5.1.6 VF Mailbox Memory — VFMBMEM (0x00200 + 4*n, n=0...15; RW)

Mailbox memory for PF and VF drivers communication. The mailbox size for each VM is 64 bytes accessed by32-bit registers. Locations can be accessed as 32-bit or 64-bit words.

Field	Bit(s)	Init Val	Description
Mailbox Data	31:0	X	*Mailbox Data* field composed of 16 x 4 byte registers.



8.3.5.1.7 VF Rx Packet Buffer Flush Detect — VFRXMEMWRAP (0x03190; RO)

This register mirrors the PF RXMEMWRAP described in Section 8.2.3.8.11.

8.3.5.2 Interrupt Registers — VF

8.3.5.2.1 VF Extended Interrupt Cause — VFEICR (0x00100; RC/W1C)

Field	Bit(s)	Init Val	Description
MSIX	2:0	0x0	Indicates an interrupt cause mapped to MSI-X vectors 2:0.
Reserved	31:3	0x0	Reserved

8.3.5.2.2 VF Extended Interrupt Cause Set — VFEICS (0x00104; WO)

Field	Bit(s)	Init Val	Description
MSIX	2:0	0x0	Sets to corresponding *EICR* bit of MSI-X vectors 2:0.
Reserved	31:3	0x0	Reserved

8.3.5.2.3 VF Extended Interrupt Mask Set/Read — VFEIMS (0x00108; RWS)

Field	Bit(s)	Init Val	Description
MSIX	2:0	0x0	*Set Mask* bit for the corresponding *EICR* bit of MSI-X vectors 2:0.
Reserved	31:3	0x0	Reserved

8.3.5.2.4 VF Extended Interrupt Mask Clear — VFEIMC (0x0010C; WO)

Field	Bit(s)	Init Val	Description
MSIX	2:0	0x0	*Clear Mask* bit for the corresponding *EICR* bit of MSI-X vectors 2:0.
Reserved	31:3	0x0	Reserved

8.3.5.2.5 VF Extended Interrupt Auto Mask Enable — VFEIAM (0x00114; RW)

Field	Bit(s)	Init Val	Description
MSIX	2:0	0x0	*Auto Mask* bit for the corresponding *EICR* bit of MSI-X vectors 2:0.
Reserved	31:3	0x0	Reserved

8.3.5.2.6 VF Extended Interrupt Mask Set/Read — VFEITR[n] (0x00820 + 4*n, n=0...1; RWS)

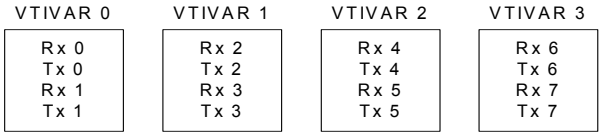
See register description in Section 8.2.3.5.12.

8.3.5.2.7 VF Interrupt Vector Allocation Registers — VFIVAR[n] (0x00120 + 4*n, n=0...3; RW)

Field	Bit(s)	Init Val	Description
INT_Alloc[0]	0	X	Defines the MSI-X vector (0 or 1) assigned to Rx queue ‘2*N’ for IVAR ‘N’ register (N=0...3).
reserved	6:1	0x0	Reserved.
INT_Alloc_val[0]	7	0b	Valid bit for INT_Alloc[0].
INT_Alloc[1]	8	X	Defines the MSI-X vector (0 or 1) assigned to Tx queue ‘2*N’ for IVAR ‘N’ register (N=0...3).
reserved	14:9	0x0	Reserved.
INT_Alloc_val[1]	15	0b	Valid bit for INT_Alloc[1].
INT_Alloc[2]	16	X	Defines the MSI-X vector (0 or 1) assigned to Rx queue ‘2*N+1’ for IVAR ‘N’ register (N=0...3).
reserved	22:17	0x0	Reserved.
INT_Alloc_val[2]	23	0b	Valid bit for INT_Alloc[2].
INT_Alloc[3]	24	X	Defines the MSI-X vector (0 or 1) assigned to Tx queue ‘2*N+1’ for IVAR ‘N’ register (N=0...3).
reserved	30:25	0x0	Reserved.
INT_Alloc_val[3]	31	0b	Valid bit for INT_Alloc[3].

These registers map interrupt causes into MSI-X vectors. See additional details in Section 7.3.4.

Transmit and receive queues mapping to VFIVAR registers is as follows:



8.3.5.2.8 VF Interrupt Vector Allocation Registers — VFIVAR_MISC (0x00140; RW)

This register maps the mailbox interrupt into an MSI-X vector. See additional details in Section 7.3.4.

Field	Bit(s)	Init Val	Description
INT_Alloc[0]	1:0	X	Defines the MSI-X vector assigned to the mailbox interrupt.
Reserved	6:2	0x0	Reserved.
INT_Alloc_val[0]	7	0b	Valid bit for INT_Alloc[0].
Reserved	31:8	0x0	Reserved.

8.3.5.2.9 VF RSC Enable Interrupt — VFRSCINT[n] (0x00180 + 4*n, n=0,1; RW)

See register description in Section 8.2.3.5.12.

8.3.5.2.10 VF MSI—X PBA Clear — VFPBACL (0x00148; RW1C)

Field	Bit(s)	Init Val	Description
PENBIT	2:0	000b	MSI-X Pending Bits Clear. Writing a 1b to any bit clears the corresponding MSIXPBA bit; writing a 0x0 has no effect. Reading this register returns the PBA vector.
Reserved	31:3	0x0	Reserved.

8.3.5.3 Receive DMA Registers — VF

8.3.5.3.1 VF Receive Descriptor Base Address Low — VFRDBAL[n] (0x01000 + 0x40*n, n=0...7; RW)

See RDBAL description in Section 8.2.3.8.1.

8.3.5.3.2 VF Receive Descriptor Base Address High — VFRDBAH[n] (0x01004 + 0x40*n, n=0...7; RW)

See RDBAH description in Section 8.2.3.8.2.

8.3.5.3.3 VF Receive Descriptor Ring Length — VFRDLEN[n] (0x01008 + 0x40*n, n=0...7; RW)

See RDLEN description in Section 8.2.3.8.3.

8.3.5.3.4 VF Receive Descriptor Head — VFRDH[n] (0x01010 + 0x40*n, n=0...7; RO)

See RDH description in Section 8.2.3.8.4.

8.3.5.3.5 VF Receive Descriptor Tail — VFRDT[n] (0x01018 + 0x40*n, n=0...7; RW)

See RDT description in Section 8.2.3.8.5.

8.3.5.3.6 VF Receive Descriptor Control — VFRXDCTL[n] (0x01028 + 0x40*n, n=0...7; RW)

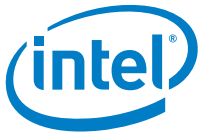
See RXDCTL description in Section 8.2.3.8.6.

8.3.5.3.7 VF Split and Replication Receive Control Register queue — VFSRRCTL (0x01014 + 0x40*n, n=0...7; RW)

See SRRCTL description in Section 8.2.3.8.7.

8.3.5.3.8 VF Replication Packet Split Receive Type — VFPSRTYPE (0x00300; RW)

See PSRTYPE description in Section 8.2.3.7.4.



8.3.5.3.9 VF RSC Control — VFRSCCTL[n] (0x0102C + 0x40*n, n=0...7; RW)

See RSCCTL description in Section 8.2.3.8.13.

8.3.5.4 Transmit Registers — VF

8.3.5.4.1 VF Transmit Descriptor Base Address Low — VFTDBAL[n] (0x02000 + n*0x40, n=0...3; RW)

See TDBAL description in Section 8.2.3.9.5.

8.3.5.4.2 VF Transmit Descriptor Base Address High — VFTDBAH[n] (0x02004 + n*0x40, n=0...3; RW)

See TDBAH description in Section 8.2.3.9.6.

8.3.5.4.3 VF Transmit Descriptor Ring Length — VFTDLEN[n] (0x02008 + n*0x40, n=0...3; RW)

See TDLEN description in Section 8.2.3.9.7.

8.3.5.4.4 VF Transmit Descriptor Head — VFTDH[n] (0x02010 + n*0x40, n=0...3; RO)

See TDH description in Section 8.2.3.9.8.

8.3.5.4.5 VF Transmit Descriptor Tail — VFTDT[n] (0x02018 + n*0x40, n=0...3; RW)

See TDT description in Section 8.2.3.9.9.

8.3.5.4.6 VF Transmit Descriptor Control — VFTXDCTL[n] (0x02028 + n*0x40, n=0...3; RW)

See RSCCTL description in Section 8.2.3.9.10.

8.3.5.4.7 VF Tx Descriptor Completion Write-Back Address Low — VFTDWBAL[n] (0x02038 + n*0x40, n=0...3; RW)

See RSCCTL description in Section 8.2.3.9.11.

8.3.5.4.8 VF Tx Descriptor Completion Write-Back Address High — VFTDWBAH[n] (0x0203C + n*0x40, n=0...3;RW)

See RSCCTL description in Section 8.2.3.9.12.

8.3.5.5 DCA Registers — VF

8.3.5.5.1 Rx DCA Control Registers — VFDCA_RXCTRL[n] (0x0100C + 0x40*n, n=0...7; RW)

See DCA_RXCTRL description in Section 8.2.3.11.1.

8.3.5.5.2 Tx DCA Control Registers — VFDCA_TXCTRL[n] (0x0200C + 0x40*n, n=0...7; RW)

See DCA_TXCTRL description in Section 8.2.3.11.2.

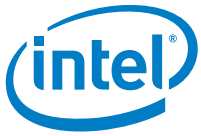
8.3.5.6 Statistic Register Descriptions — VF

8.3.5.6.1 VF Good Packets Received Count — VFGPRC (0x0101C; RO)

Field	Bit(s)	Init Val	Description
GPRC	31:0	0x0	Number of good packets received for this VF (of any length). This counter includes loopback packets or replications of multicast packets. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000.

8.3.5.6.2 VF Good Packets Transmitted Count — VFGPTC (0x0201C; RO)

Field	Bit(s)	Init Val	Description
GPTC	31:0	0x0	Number of good packets sent by the queues allocated to this VF. This counter includes loopback packets or packets latter dropped by the switch or the MAC but does not include packet dropped by anti spoofing or VLAN tag filtering (as described in Section 7.10.3.9.2). The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000.



8.3.5.6.3 VF Good Octets Received Count Low — VFGORC_LSB (0x01020; RO)

Field	Bit(s)	Init Val	Description
GORC-LSB	31:0	0x0	Number of good octets received (32 LS bits of a 36-bit counter) by the queues allocated to this VF. The counter includes loopback packets or replications of multicast packets. This register includes bytes received in a packet from the <Destination Address> field through the <CRC> field, inclusively. Octets are counted on the VF interface rather than on the network interface (such as LinkSec octets not being counted). Bytes of RSC are counted before coalescing. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000.

8.3.5.6.4 VF Good Octets Received Count High — VFGORC_MSB (0x01024; RO)

Field	Bit(s)	Init Val	Description
GORC-MSB	3:0	0x0	Number of good octets received (4 MS bits of a 36-bit counter) by the queues allocated to this VF. See the complete explanation in Section 8.3.5.6.3. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xF to 0x0.

8.3.5.6.5 VF Good Octets Transmitted Count — VFGOTC_LSB (0x02020; RO)

Field	Bit(s)	Init Val	Description
GOTC-LSB	31:0	0x0	Number of good octets transmitted (32 LS bits of a 36-bit counter) by the queues allocated to this VF. This register includes bytes transmitted in a packet from the <Destination Address> field through the <CRC> field, inclusively. This register counts octets of the packets counted by the VFGPTC register. Octets are counted on the VF interface rather than on the network interface (such as LinkSec octets not being counted). The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000.

8.3.5.6.6 VF Good Octets Transmitted Count — VFGOTC_MSB (0x02024; RO)

Field	Bit(s)	Init Val	Description
GOTC-MSB	3:0	0x0	Number of good octets transmitted (4 MS bits of a 36-bit counter) by the queues allocated to this VF. See the complete explanation in Section 8.3.5.6.5. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xF to 0x0.
Reserved	31:4	0x0	Reserved.

8.3.5.6.7 VF Multicast Packets Received Count — VFMPRC (0x01034; RO)

Field	Bit(s)	Init Val	Description
MPRC	31:0	0x0	Number of multicast good packets received by this VF (of any length) that pass Ethernet MAC address filtering (excluding broadcast packets). The counter does not count received flow control packets. This register increments only if receives are enabled. This register does not count packets counted by the Missed Packet Count (MPC) register. This counter includes loopback packets or replications of multicast packets. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000.

9.0 PCIe Programming Interface

9.1 PCI Compatibility

PCIe is fully compatible with existing deployed PCI software. To achieve this, PCIe hardware implementations conform to the following requirements:

- All devices are required to be supported by deployed PCI software and must be enumerable as part of a tree-through PCI device enumeration mechanisms.
- Devices must not require any resources such as address decode ranges and interrupts beyond those claimed by PCI resources for operation of software compatible and software transparent features with respect to existing deployed PCI software.
- Devices in their default operating state must confirm to PCI ordering and cache coherency rules from a software viewpoint.
- PCIe devices must conform to the PCI power management specification and must not require any register programming for PCI-compatible power management beyond those available through PCI power management capability registers. Power management is expected to conform to a standard PCI power management by existing PCI bus drivers.

PCIe devices implement all registers required by the PCI specification as well as the power management registers and capability pointers specified by the PCI power management specification. In addition, PCIe defines a PCIe capability pointer to indicate support for PCIe extensions and associated capabilities.

The 82599 is a multi-function device with the following functions:

- LAN 0
- LAN 1

Different parameters affect how LAN functions are exposed on PCIe.

Both functions contain the following regions of the PCI configuration space (some of them are enabled by EEPROM settings as detailed in the following sections):

- Mandatory PCI configuration registers
- Power management capabilities
- MSI / MSI-X capabilities
- Vital Product Data (VPD) capability

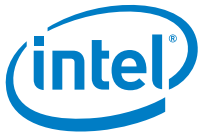
- PCIe extended capabilities:
  - Advanced Error Reporting (AER)
  - Serial ID
  - Alternate requester ID.
  - Single root IOV

9.2 Configuration Sharing Among PCI Functions

The 82599 contains a single physical PCIe core interface. It is designed so that each of the logical LAN devices (LAN 0, LAN 1) appears as a distinct function implementing its own PCIe device header space.

Many of the fields of the PCIe header space contain hardware default values that are either fixed or can be overridden using an EEPROM, but might not be independently specified for each logical LAN device. The following fields are considered to be common to both LAN functions:

Vendor ID	The Vendor ID of the 82599 is specified to a single value 0x8086. The value is reflected identically for both LAN devices.
Revision	The revision number of the 82599 is reflected identically for both LAN devices.
Header Type	This field indicates if a device is single function or multi-function. The value reflected in this field is reflected identically for both LAN devices, but the actual value reflected depends on LAN disable configuration. When both the 82599 LAN ports are enabled, both PCIe headers return 0x80 in this field, acknowledging being part of a multi-function device. LAN 0 exists as device function 0, while LAN 1 exists as device function 1. If function 1 is disabled, then only a single-function device is indicated (this field returns a value of 0x00) and the LAN exists as device function 0.
Subsystem ID	The Subsystem ID of the 82599 can be specified via an EEPROM, but only a single value can be specified. The value is reflected identically for both LAN devices.
Subsystem Vendor ID	The Subsystem Vendor ID of the 82599 can be specified via an EEPROM, but only a single value can be specified. The value is reflected identically for both LAN devices.
Cap_Ptr Max Latency Min Grant	These fields reflect fixed values that are constant values reflected for both LAN devices.



The following fields are implemented as unique to each LAN functions:

Device ID	The Device ID reflected for each LAN function can be independently specified via an EEPROM.
Command Status	Each LAN function implements its own Command/Status registers.
Latency Timer Cache Line Size	Each LAN function implements these registers independently. The system should program these fields identically for each LAN to ensure consistent behavior and performance of each device.
Memory BAR, IO BAR Expansion ROM BAR MSIX BAR	Each LAN function implements its own Base Address registers, enabling each device to claim its own address region(s). The I/O BAR is enabled by the *IO_Sup* bit in the EEPROM.
Interrupt Pin	Each LAN function independently indicates which interrupt pin (INTA#...INTD#) is used by that device's MAC to signal system interrupts. The value for each LAN device can be independently specified via an EEPROM, but only if both LAN devices are enabled.
Class Code	Each function can have its own device class. Function 0 can be dummy function, LAN or storage and Function 1 can be either LAN or storage. Both are enabled by the EEPROM.

9.3 PCIe Register Map

Configuration registers are assigned one of the attributes described in the table that follows.

9.3.1 Register Attributes

The following table lists the register attributes used in this section.

RD/WR	Description
RO	Read-only register: Register bits are read-only and cannot be altered by software.
RW	Read-write register: Register bits are read-write and can be either set or reset.
RW1C	Read-only status, Write-1b-to-clear status register, Writing a 0b to RW1C bits has no effect.
ROS	Read-only register with sticky bits: Register bits are read-only and cannot be altered by software. Bits are not cleared by reset and can only be reset with the PWRGOOD signal. Devices that consume AUX power are not allowed to reset sticky bits when AUX power consumption (either via AUX power or PME Enable) is enabled.
RWS	Read-write register: Register bits are read-write and can be either set or reset by software to the desired state. Bits are not cleared by reset and can only be reset with the PWRGOOD signal. Devices that consume AUX power are not allowed to reset sticky bits when AUX power consumption (either via AUX power or PME Enable) is enabled.
RW1CS	Read-only status, Write-1b-to-clear status register: Register bits indicate status when read, a set bit, indicating a status event, can be cleared by writing a 1b to it. Writing a 0b to RW1C bits has no effect. Bits are not cleared by reset and can only be reset with the PWRGOOD signal. Devices that consume AUX power are not allowed to reset sticky bits when AUX power consumption (either via AUX power or PME Enable) is enabled.
HwInit	Hardware initialized: Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial EEPROM. Bits are read-only after initialization and can only be reset (for write-once by firmware) with the PWRGOOD signal.
RsvdP	Reserved and preserved: Reserved for future read/write implementations; software must preserve value read for writes to these bits.
RsvdZ	Reserved and zero: Reserved for future RW1C implementations; software must use 0b for writes to these bits.

9.3.2 PCIe Configuration Space Summary

Table 9-1lists the PCIe configuration registers while their detailed description is given in the sections that follow. PCI configuration fields in the summary table are presented by the following marking:

- Fields that have meaningful default values are indicated in parenthesis — (**value**).
- Dotted fields indicates the same value for both LAN functions
- Light-blue fields indicate read-only fields (loaded from the EEPROM)
- Magenta fields indicate hard-coded values.

- Other fields contain RW attributes.

Table 9-1 PCI Configuration Space

Section	Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
Mandatory PCI Register	0x0	Device ID		Vendor ID	
	0x4	Status Register		Control Register	
	0x8	Class Code (0x020000/0x010000)			Revision ID
	0xC	Reserved	Header Type (0x0/ 0x80)	Latency Timer	Cache Line Size (0x10)
	0x10	Base Address Register 0			
	0x14	Base Address Register 1			
	0x18	Base Address Register 2			
	0x1C	Base Address Register 3			
	0x20	Base Address Register 4			
	0x24	Base Address Register 5			
	0x28	CardBus CIS pointer (0x0000)			
	0x2C	Subsystem ID		Subsystem Vendor ID	
	0x30	Expansion ROM Base Address			
	0x34	Reserved			Cap Ptr (0x40)
	0x38	Reserved			
	0x3C	Max Latency (0x00)	Min Grant (0x00)	Interrupt Pin (0x01...0x04)	Interrupt Line (0x00)
PCI / PCIe Capabilities	0x40...0x47	Power management capability			
	0x50...0x67	MSI Capability			
	0x70...0x7B	MSI-X Capability			
	0xA0...0xDB	PCIe Capability			
	0xE0...0xE7	VPD Capability			
Extended PCIe Configuration	0x100...0x12B	AER Capability			
	0z140...0x14B	Serial ID Capability			
	0x150...0x157	ARI Capability			
	0x160...0x19C	SR-IOV Capability			

9.3.3 Mandatory PCI Configuration Registers — Except BARs

9.3.3.1 Vendor ID Register (0x0; RO)

This is a read-only register that has the same value for all PCI functions. It identifies unique Intel products.

9.3.3.2 Device ID Register (0x2; RO)

This is a read-only register that identifies individual the 82599 PCI functions. Both ports have the same default value equals to 0x10D8, and can be auto-loaded from the EEPROM during initialization with different values for each port as well as the dummy function (See Section 4.4 for dummy function relevance).

9.3.3.3 Command Register (0x4; RW)

Shaded bits are not used by this implementation and are hardwired to 0b. Each function has its own Command register. Unless explicitly specified, functionality is the same in both functions.

Bit(s)	Init Val	Description
0	0b	I/O Access Enable.
1	0b	Memory Access Enable.
2	0b	Enable Mastering, also named Bus Master Enable (BME)). • LAN functions RW field • Dummy function RO as zero field
3	0b	Special Cycle Monitoring – Hardwire to 0b.
4	0b	MWI Enable – Hardwire to 0b.
5	0b	Palette Snoop Enable – Hardwire to 0b.
6	0b	Parity Error Response.
7	0b	Wait Cycle Enable – Hardwired to 0b.
8	0b	SERR# Enable.
9	0b	Fast Back-to-Back Enable – Hardwire to 0b.
10	1b	Interrupt Disable. When set, devices are prevented from generating legacy interrupt messages.
15:11	0b	Reserved.