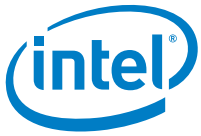


If DCB is enabled then following a software reset the following steps must be executed to prevent potential races between manageability mapping to TC before and after initialization.

1. Clear the flow control enablement in the MAC by clearing MFLCN.RFCE (or clear the entire register).
2. Software should wait ~10 µs.
3. Software polls TFCS.TC_XON(0) = 0b (in most cases it is expected to be found at zero while max poll time is always shorter than the max expected PAUSE time before a software reset is initiated).
4. Software maps the manageability transmit TC (setting the MNGTXMAP register) and then maps the user priority of manageability traffic to the manageability TC (setting the RTRUP2TC and RTTUP2TC registers).
5. Software waits ~10 µs.
6. Software can re-enable the flow control as part of the rest of the initialization flow.

4.2.1.6.2 Physical Function (PF) Software Reset

A software reset by the PF in IOV mode has the same consequences as a software reset in non-IOV mode.

The procedure for a PF software reset is as follows:

- The PF driver disables master accesses by the device through the master disable mechanism (see Section 5.2.5.3.2). Master disable affects all VFs traffic.
- Execute the procedure described in Section 4.2.2 to synchronize between the PF and VFs.

VFs are expected to timeout and check on the *RSTD* bit in order to identify a PF software reset event. The *RSTD* bits are cleared on read.

4.2.1.6.3 VF Software Reset

A software reset applied by a VF is equivalent to a FLR reset to this VF with the exception that the PCIe configuration bits allocated to this function are not reset. It is activated by setting the VTCTRL.RST bit.

4.2.1.6.4 Force TCO

This reset is generated when manageability logic is enabled. It is only generated if enabled by the *Force TCO Reset* bit in the Common Firmware Parameters word in the EEPROM. If enabled by the EEPROM, firmware triggers a port reset by setting the CTRL.RST bit. In pass through mode it is generated when receiving a ForceTCO SMB command with bit 0 set.

4.2.1.7 Link Reset

Also referred to as MAC reset.

Initiated by writing the *Link Reset* bit of the Device Control register (CTRL.LRST).

A link reset is equivalent to a software reset + reset of the MAC. The 82599 re-reads the per-function EEPROM fields after link reset. Bits that are normally read from the EEPROM are reset to their default hardware values. Note that this reset is per function and resets only the function that received the link reset.

The PF in IOV mode can also generate a link reset.

Prior to issuing link reset, the driver needs to execute the master disable algorithm as defined in Section 5.2.5.3.2.

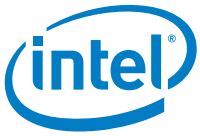
4.2.2 Reset in PCI-IOV Environment

Several mechanisms are provided to synchronize reset procedures between the PF and the VFs.

4.2.2.1 (RSTI)/(RSTD)

This mechanism is provided specifically for a PF software reset but can be used in other reset cases. The procedure is as follows:

- One of the following reset cases takes place:
  - LAN Power Good
  - PCIe Reset (PERST and in-band)
  - D3hot --> D0
  - FLR
  - Software reset by the PF
- The 82599 sets the *RSTI* bits in all the VFMailbox registers. Once the reset completes, each VF can read its VFMailbox register to identify a reset in progress.
  - The VF might poll the *RSTI* bit to detect if the PF is in the process of configuring the device.
- Once the PF completes configuring the device, it sets the CTRL_EXT.PFRSTD bit. As a result, the 82599 clears the *RSTI* bits in all the VFMailbox registers and sets the Reset Done (*RSTD*) bits in all the VFMailbox registers.
  - The VF might read the *RSTD* bit to detect that a reset has occurred. The *RSTD* bit is cleared on read.



4.2.2.2 VF Receive Enable — PFVFRE / VF Transmit Enable — PFVFTE

This mechanism insures that a VF cannot transmit or receive before the Tx and Rx path has been initialized by the PF.

- The PFVFRE register contains a bit per VF. When the bit is set to 0b, Rx packet assignment for the VF's pool is disabled. When set to 1b, Rx packet assignment for the VF's pool is enabled.
- The PFVFTE register contains a bit per VF. When the bit is set to 0b, data fetching for the VF's pool is disabled. When set to 1b, data fetching for the VF's pool is enabled. Descriptor fetching for the VF pool is maintained, up to the limit of the internal descriptor queues — regardless of PFVFTE settings.

The PFVFTE and PFVFRE registers are initialized to zero (VF Tx and Rx traffic gated) following a PF reset. The relevant bits per VF are also initialized by a VF software reset or VFLR.

4.2.3 Reset Effects

Table 4-4 through Table 4-6 list how resets affect the following registers and logic:

Table 4-4 Reset Effects — Common Resets

Reset Activation	LAN Power Good	PCIe PERST#	In-band PCIe Reset	FW Reset	Force TCO	Notes
EEPROM Read	See Section 6.3.1					
LTSSM (back to detect/polling)	X	X	X			
PCIe Link Data Path	X	X	X			
PCI Configuration Registers RO	X	X	X			9
PCI Configuration Registers RW	X	X	X			9
PCIe Local Registers	X	X	X			8
Data Path	X	X	X		X	2
On-die Memories	X	X	X		X	7
MAC, PCS, Auto-Negotiation, LinkSec, IPsec	X	X 6	X 6		X	
Wake Up (PM) Context	X	1				3
Wake Up/Manageability Control/Status Regs	X					4, 5

Table 4-4 Reset Effects — Common Resets (Continued)

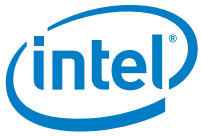
Reset Activation	LAN Power Good	PCIe PERST#	In-band PCIe Reset	FW Reset	Force TCO	Notes
Manageability Unit	X			X		
LAN Disable Strapping Pins	X					
All Other Strapping Pins	X					

Table 4-5 Reset Effects — Per Function Resets

Reset Activation	D3 or Dr	FLR or PFLR	SW Reset	Link Reset or Exit from LAN Disable	Notes
EEPROM Read	See Section 6.3.1				
LTSSM (back to detect/polling)					
PCIe Link Data Path					
PCI Configuration Registers RO					9
PCI Configuration Registers RW	X	X			9
Data path, Memory Space	X	X	X	X	2
On-die Memories	X	X	X		7
MAC, PCS, Auto-Negotiation, LinkSec, IPsec	X 6	X 6	X 6	X	
Virtual Function Resources	X	X	X		10
Wake Up (PM) Context					3
Wake Up/Manageability Control/Status Regs					4,5
Manageability Unit					
Strapping Pins					

Table 4-6 Reset Effects -Virtual Function Resets

Reset Activation	VFLR	VF SW Reset	Notes
Interrupt Registers	X	X	11
Queue Disable	X	X	12
VF Specific PCIe Configuration Space	X		13
Data Path			
Statistics Registers			14



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. Refer to Section 8.3.5.1.5 for more details.

Notes For Table 4-4 through Table 4-6:

1. If AUX_PWR = 0b the wake up context is reset (*PME_Status* and *PME_En* bits should be 0b at reset if the 82599 does not support PME from D3cold).
2. The following register fields do not follow the previous general rules:
   - ESDP registers- reset on LAN Power Good only.
   - LED configuration registers.
   - The *Aux Power Detected* bit in the PCIe Device Status register is reset on LAN Power Good and PCIe Reset only.
   - FLA — reset on LAN Power Good only.
   - RAH/RAL[n, where n>0], MTA[n], VFTA[n], FHFT_*[n], TDBAH/TDBAL, and RDBAH/RDVAL registers have no default value. If the functions associated with the registers are enabled they must be programmed by software. Once programmed, their value is preserved through all resets as long as power is applied.
   - Statistic registers (physical function)
3. The wake up context is defined in the PCI Bus Power Management Interface Specification (sticky bits). It includes:
   - *PME_En* bit of the Power Management Control/Status Register (PMCSR)
   - *PME_Status* bit of the Power Management Control/Status Register (PMCSR)
   - Aux_En in the PCIe registers
   - The device requester ID (since it is required for the PM_PME TLP)

   The shadow copies of these bits in the Wakeup Control Register are treated identically.
4. Refers to bits in the Wake Up Control Register that are not part of the Wake-Up Context (the PME_En and PME_Status bits). Note that the WUFC and WUC registers are not part of the Wake Up Context and are reset as part of the data path. Include also the SW_FW_SYNC and the FWSM registers.
5. The Wake Up Status Registers include the following:
   - Wake Up Status Register
   - Wake Up Packet Length
   - Wake Up Packet Memory
6. The MAC cluster is reset by the appropriate event only if manageability unit is disabled and the host is in a low power state with WoL disabled.
7. The contents of the following memories are cleared to support the requirements of PCIe FLR:
   - The Tx packet buffers
   - The Rx packet buffers
   - IPsec Tx SA tables
   - IPsec Rx SA tables
8. The following registers are part of this group:
   - SWSM
   - GCR (only bit 9 is cleared by this reset while all other fields are cleared at LAN Power Good reset)
   - GSCL_1/GSCL_2
   - GSCN_0/1/2/3
9. Sticky bits and hardware init bits (indicated as HwInit) in the PCI Configuration registers are cleared only by LAN Power Good reset.
10. These registers include:
    - VFEICS
    - VFEIMS
    - VFEIAC
    - VFEIAM
    - VFEITR 0-2
    - VTIVAR0
    - VFIVAR_MISC
    - VFPBACL
    - VFMailbox
11. These registers include:
    - VFEICS

- VFEIMS
- VFEIMC
- VFEIAC
- VFEIAM
- VFEICR
- EITR 0-2
- VTIVAR0
- VFIVAR_MISC
- VFPBACL
- VFMailbox

12. These registers include:
    - Specific VF bits in the FVRE and FVTE are cleared as well
13. These registers include:
    - MSI/MSI-X enable bits
    - BME
    - Error indications
14. Rx and Tx counters might miss proper counting due to VFLR indicating more packets than those ones actually transferred. It could happen if the VFLR happened after counting occurred but before Tx or Rx were completed.

4.3 Queue Disable

See Section 4.6.7.1 for details on disabling and enabling an Rx queue.

See Section 4.6.8.1 for details on disabling and enabling a Tx queue.

4.4 Function Disable

4.4.1 General

For a LAN on Motherboard (LOM) design, it might be desirable for the system to provide BIOS-setup capability for selectively enabling or disabling LAN functions. It enables the end-user more control over system resource-management and avoids conflicts with add-in NIC solutions. The 82599 provides support for selectively enabling or disabling one or both LAN device(s) in the system.

4.4.2 Overview

Device presence (or non-presence) must be established early during BIOS execution, in order to ensure that BIOS resource-allocation (of interrupts, of memory or IO regions) is done according to devices that are present only. This is frequently accomplished using a BIOS Configuration Values Driven on Reset (CVDR) mechanism. The 82599 LAN-disable mechanism is implemented in order to be compatible with such a solution.

The 82599 provides two mechanisms to disable each of its LAN ports:

- The LANx_DIS_N pins (one pin per LAN port) are sampled on reset to determine the LAN enablement.
- One of the LAN ports can be disabled using EEPROM configuration.

Disabling a LAN port affects the PCI function it resides on. When function 0 is disabled (either LAN0 or LAN1), two different behaviors are possible:

- Dummy function mode — In some systems, it is required to keep all the functions at their respective location, even when other functions are disabled. In dummy function mode, if function #0 (either LAN0 or LAN1) is disabled, then it does not disappear from the PCIe configuration space. Rather, the function presents itself as a dummy function. The device ID and class code of this function changes to other values (dummy function device ID 0x10A6 and class code 0xFF0000). In addition, the function does not require any memory or I/O space, and does not require an interrupt line.
- Legacy mode — When function 0 is disabled (either LAN0 or LAN1), then the port residing on function 1 moves to reside on function 0. Function 1 disappears from the PCI configuration space.

Mapping between function and LAN ports is listed in the following tables.

Table 4-7 PCI Functions Mapping (Legacy Mode)

PCI Function #	LAN Function Select	Function 0	Function 1
Both LAN functions are enabled.	0	LAN 0	LAN 1
	1	LAN 1	LAN 0
LAN 0 is disabled.	x	LAN1	Disable
LAN 1 is disabled.	x	LAN 0	Disable
Both LAN functions are disabled.	Both PCI functions are disabled. Device is in low power mode.		

Table 4-8 PCI Functions Mapping (Dummy Function Mode)

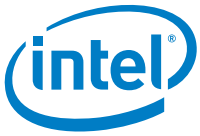
PCI Function #	LAN Function Select	Function 0	Function 1
Both LAN functions are enabled.	0	LAN 0	LAN 1
	1	LAN 1	LAN 0
LAN 0 is disabled.	0	Dummy	LAN1
	1	LAN 1	Disable
LAN 1 is disabled.	0	LAN 0	Disable
	1	Dummy	LAN 0
Both LAN functions are disabled.	Both PCI functions are disabled. Device is in low power mode.		

The following rules apply to function disable:

- When function 0 is disabled in legacy mode, the LAN port associated originally with function 1 appears in function 0. Function 1 disappears from the PCI configuration space.
- When function 0 is disabled in dummy function mode, it is converted into a dummy PCI function. Function 1 is not affected.
- When function 1 is disabled, it disappears from the PCI configuration space.
- The disabled LAN port is still available for manageability purposes if disabled through the EEPROM mechanism. The disabled LAN port is not available for manageability purposes if disabled through the pin mechanism.
- Dummy function mode should not be used in PCI IOV mode (since PF0 is required to support certain functionality)

The following EEPROM bits control function disable:

- One PCI function can be enabled or disabled according to the EEPROM *LAN PCI Disable* bit.
- The *LAN Disable Select* EEPROM field indicates which function is disabled.
- The *LAN Function Select* EEPROM bit defines the correspondence between LAN Port and PCI function
- The *Dummy Function Enable* EEPROM bit enables the dummy function mode. Default value is disabled.



4.4.3 Control Options

The functions have a separate enabling mechanism. Any function that is not enabled does not function and does not expose its PCI configuration registers.

LAN0 **or** LAN 1 can be disabled in the EEPROM by setting the *LAN PCI Disable* bit in the PCIe Control 2 word at offset 0x05. The *LAN Disable Select* bit in the same word in the EEPROM selects which LAN is disabled. Furthermore, if the LAN port at function 0 is disabled, the *Dummy Function Enable* bit in the same word chooses between filling the disabled function by a dummy function, or moving the other LAN port to function 0.

Note: Mapping LAN0 and LAN1 to PCI function 0 and PCI function 1 is controlled by the EEPROM *LAN Function Select* bit in the PCIe Control 2 word at offset 0x05.

LAN0 **and** LAN 1 can be disabled on the board level by driving the LAN0_Dis_N and LAN1_Dis_N pins to low. These I/O pins have internal weak pull-up resistors so leaving them unconnected or driving them to high enables the respective LAN port. These pins are strapping options, sampled at LAN Power Good, PCIe reset or in-band PCIe reset.

4.4.4 Event Flow for Enable/Disable Functions

This section describes the driving levels and event sequence for device functionality. Following a Power on Reset / LAN Power Good/ PCIe Reset/ In-Band Reset, the LANx_DIS_N signals should be driven high (or left open) for normal operation. If any of the LAN functions are not required statically, its associated disable strapping pin can be tied statically to low.

4.4.4.1 BIOS Disable the LAN Function at Boot Time by Using Strapping Option

Assume that following a power up sequence LANx_DIS_N signals are driven high.

1. PCIe is established following PCIe reset.
2. BIOS recognizes that a LAN function in the 82599 should be disabled.
3. The BIOS drives the LANx_DIS_N signal to the low level.
4. BIOS issues PCIe reset or an in-band PCIe reset.
5. As a result, the 82599 samples the LANx_DIS_N signals and disables the LAN function and issues an internal reset to this function.
6. BIOS might start with the device enumeration procedure (the disabled LAN function is invisible — changed to dummy function).
7. Proceed with normal operation.
8. Re-enable could be done by driving the LANx_DIS_N signal high and then requesting the user to issue a warm boot to initialize new bus enumeration.

4.4.4.2 Multi-Function Advertisement

If one of the LAN devices is disabled and function 0 is the only active function, the 82599 is no longer a multi-function device. The 82599 normally reports a 0x80 in the PCI configuration header, indicating multi-function capability. However, if a LAN is disabled and only function 0 is active, the 82599 reports a 0x0 in this field to signify single-function capability.

4.4.4.3 Interrupt Utilization

When both LAN devices are enabled, the 82599 uses the PCI legacy interrupts of both ports for interrupt reporting. The EEPROM configuration controls the *Interrupt Pin* field of the PCI configuration header to be advertised for each LAN device to comply with PCI specification requirements.

However, if either LAN device is disabled, then the legacy PCI interrupt of port A must be used for the remaining LAN device, therefore the EEPROM configuration must be set accordingly. Under these circumstances, the *Interrupt Pin* field of the PCI configuration header always reports a value of 0x1, indicating INTA# pin usage, which means legacy PCI interrupt of port A is used.

4.4.4.4 Power Reporting

When both LAN devices are enabled, the PCI Power Management register block has the capability of reporting a common power value. The common power value is reflected in the *Data* field of the PCI Power Management registers. The value reported as common power is specified via an EEPROM field, and is reflected in the *Data* field each time the *Data_Select* field has a value of 0x8 (0x8 = common power value select).

When only one LAN port is enabled and the 82599 appears as a single-function device, the common power value, if selected, reports 0x0 (undefined value), as common power is undefined for a single-function device.

4.5 Device Disable

4.5.1 Overview

When both LAN ports are disabled following a Power on Reset / LAN Power Good/ PCIe Reset/ In-Band Reset, the LANx_DIS_N signals should be tied statically to low. In this state the device is disabled, LAN ports are powered down, all internal clocks are shut, and the PCIe connection is powered down (similar to L2 state).

4.5.2 BIOS Disable of the Device at Boot Time by Using the Strapping Option

Assume that following power-up sequence LANx_DIS_N signals are driven high:

1. PCIe is established following PCIe reset.
2. BIOS recognizes that the 82599 should be disabled.
3. The BIOS drives the LANx_DIS_N signals to the low level.
4. BIOS issues PCIe reset or an in-band PCIe reset.
5. As a result, the 82599 samples the LANx_DIS_N signals and disables the LAN ports and the PCIe connection.
6. Re-enable can be done by driving at least one of the LANx_DIS_N signals high and then issuing a PCIe reset to restart the device.

4.6 Software Initialization and Diagnostics

4.6.1 Introduction

This section discusses general software notes for the 82599, especially initialization steps. This includes:

- General hardware power-up state
- Basic device configuration
- Initialization of transmit
- Receive operation
- Link configuration
- Software reset capability
- Statistics
- Diagnostic hints

4.6.2 Power-Up State

When the 82599 powers up, it automatically reads the EEPROM. The EEPROM contains sufficient information to bring the link up and configure the 82599 for manageability and/or APM wakeup. However, software initialization is required for normal operation.

4.6.3 Initialization Sequence

The following sequence of commands is typically issued to the device by the software device driver in order to initialize the 82599 for normal operation. The major initialization steps are:

1. Disable interrupts.
2. Issue global reset and perform general configuration (see Section 4.6.3.2).
3. Wait for EEPROM auto read completion.
4. Wait for DMA initialization done (RDRXCTL.DMAIDONE).
5. Setup the PHY and the link (see Section 4.6.4).
6. Initialize all statistical counters (see Section 4.6.5).
7. Initialize receive (see Section 4.6.7).
8. Initialize transmit (see Section 4.6.8).
9. Enable interrupts (see Section 4.6.3.1).

4.6.3.1 Interrupts During Initialization

Most drivers disable interrupts during initialization to prevent re-entrance. Interrupts are disabled by writing to the EIMC registers. Note that the interrupts also need to be disabled after issuing a global reset, so a typical driver initialization flow is:

1. Disable interrupts.
2. Issue a global reset.
3. Disable interrupts (again).

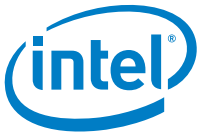
After initialization completes, a typical driver enables the desired interrupts by writing to the IMS register.

4.6.3.2 Global Reset and General Configuration

Note: Global Reset = software reset + link reset.

Device initialization typically starts with a software reset that puts the device into a known state and enables the device driver to continue the initialization sequence. Following a Global Reset the Software driver should wait at least 10msec to enable smooth initialization flow.

To enable flow control, program the FCTTV, FCRTL, FCRTH, FCRTV and FCCFG registers. If flow control is not enabled, these registers should be written with 0x0. If Tx flow control is enabled then Tx CRC by hardware should be enabled as well (HLREG0.TXCRCEN = 1b). Refer to Section 3.7.7.3.2 through Section 3.7.7.3.5 for the recommended setting of the Rx packet buffer sizes and flow control thresholds. Note that FCRTH[n].RTH fields must be set by default regardless if flow control is enabled or not. Typically, FCRTH[n] default value should be equal to RXPBSIZE[n]-0x6000. FCRTH[n].FCEN should be set to 0b if flow control is not enabled as all the other registers previously indicated.



The link interconnect configuration according to the electrical specification of the relevant electrical interface should be set prior to the link setup. This configuration is done through the EEPROM by applying the appropriate settings to the link interconnect block.

4.6.4 100 Mb/s, 1 GbE, and 10 GbE Link Initialization

4.6.4.1 BX/ SGMII Link Setup Flow

1. BX link electrical setup is done according to EEPROM configuration to set the analog interface to the appropriate setting.
2. Configure the *Link Mode Select* field in the AUTOC register to the appropriate operating mode.
3. Configure any interface fields in the SERDESC register if necessary.
4. Restart the link using the *Restart Auto Negotiation* field in the AUTOC register.
5. Verify correct link status (sync, link_up, speed) using the LINKS register.

4.6.4.2 XAUI / BX4 / CX4 / SFI Link Setup Flow

1. XAUI / BX4 / CX4 / SFI link electrical setup is done according to EEPROM configuration to set the analog interface to the appropriate setting.
2. Configure the *Link Mode Select* field in the AUTOC register, AUTOC.10G_PARALLEL_PMA_PMD and AUTOC2.10G_PMA_PMD_Serial to the appropriate operating mode.
3. Configure any interface fields in the SERDESC register if necessary.
4. Restart the link using the *Restart Auto Negotiation* field in the AUTOC register.
5. Verify correct link status (align, link_up, speed) using the LINKS register.

4.6.4.3 KX / KX4 / KR Link Setup Flow Without Auto-Negotiation

1. KX / KX4 / KR link electrical setup is done according to EEPROM configuration to set the analog interface to the appropriate setting.
2. Configure the *Link Mode Select* field in the AUTOC register, AUTOC.10G_PARALLEL_PMA_PMD and AUTOC2.10G_PMA_PMD_Serial to the appropriate operating mode.
3. Configure any interface fields in the SERDESC register if necessary.
4. Restart the link using the *Restart Auto Negotiation* field in the AUTOC register.
5. Verify correct link status (sync, align, link_up, speed) using the LINKS register.

4.6.4.4 KX / KX4 / KR Link Setup Flow With Auto-Negotiation

1. KX / KX4 / KR link electrical setup is done according to EEPROM configuration to set the analog interface to the appropriate setting.
2. Configure the *Link Mode Select* field in the AUTOC register, AUTOC.10G_PARALLEL_PMA_PMD and AUTOC2.10G_PMA_PMD_Serial to the appropriate operating mode.
3. Configure any interface fields in the SERDESC register if necessary.
4. Configure the *KX_Support* field and any other auto-negotiation related fields in the AUTOC register.
5. Restart the link using the *Restart Auto Negotiation* field in the AUTOC register.
6. Verify correct link status (sync, align, link_up, speed) using the LINKS register.

4.6.5 Initialization of Statistics

Statistics registers are hardware-initialized to values as detailed in each particular register's description. The initialization of these registers begins upon transition to D0 active power state (when internal registers become accessible, as enabled by setting the *Memory Access Enable* field of the PCIe Command register), and is guaranteed to be completed within 1 ms of this transition. Note that access to statistics registers prior to this interval might return indeterminate values.

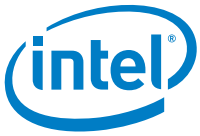
All of the statistical counters are cleared on read and a typical device driver reads them (thus making them zero) as a part of the initialization sequence.

Queue counters are mapped using the RQSMR registers for Rx queues, and TQSM registers for Tx queues. Refer to Section 8.2.3.23.71 for RQSMR setup, and Section 8.2.3.23.73 for TQSM setup. Note that if software requires the queue counters, the RQSMR and TQSM registers must be re-programmed following a device reset.

4.6.6 Interrupt Initialization

Operating with Legacy or MSI Interrupts:

- The software driver associates between Tx and Rx interrupt causes and the EICR register by setting the IVAR[n] registers.
- Program SRRCTL[n].RDMTS (per receive queue) if software uses the receive descriptor minimum threshold interrupt.
- All interrupts should be set to 0b (no auto clear in the EIAC register). Following an interrupt, software might read the EICR register to check for the interrupt causes.
- Set the auto mask in the EIAM register according to the preferred mode of operation.
- Set the interrupt throttling in EITR[n] and GPIE according to the preferred mode of operation.



- Software clears EICR by writing all ones to clear old interrupt causes.
- Software enables the required interrupt causes by setting the EIMS register.

Operating with MSI-X:

- The operating system / BIOS sets the hardware to MSI-X mode and programs the MSI-X table as part of the device enumeration procedure.
- The software driver associates between interrupt causes and MSI-X vectors and the throttling timers EITR[n] by programming the IVAR[n] and IVAR_MISC registers.
- Program SRRCTL[n].RDMTS (per receive queue) if software uses the receive descriptor minimum threshold interrupt.
- The EIAC[n] registers should be set to auto clear for transmit and receive interrupt causes (for best performance). The EIAC bits that control the other and TCP timer interrupt causes should be set to 0b (no auto clear).
- Set the auto mask in the EIAM and EIAM[n] registers according to the preferred mode of operation.
- Set the interrupt throttling in EITR[n] and GPIE according to the preferred mode of operation.
- Software enables the required interrupt causes by setting the EIMS[n] registers.

4.6.7 Receive Initialization

Initialize the following register tables before receive and transmit is enabled:

- Receive Address (RAL[n] and RAH[n]) for used addresses.
- Receive Address High (RAH[n].VAL = 0b) for unused addresses.
- Unicast Table Array (PFUTA).
- VLAN Filter Table Array (VFTA[n]).
- VLAN Pool Filter (PFVLVF[n]).
- MAC Pool Select Array (MPSAR[n]).
- VLAN Pool Filter Bitmap (PFVLVFB[n]).

Program the Receive Address register(s) (RAL[n], RAH[n]) per the station address. This can come from the EEPROM or from any other means (for example, it could be stored anywhere in the EEPROM or even in the platform PROM for LOM design).

Set up the Multicast Table Array (MTA) registers. This entire table should be zeroed and only the desired multicast addresses should be permitted (by writing 0x1 to the corresponding bit location). Set the MCSTCTRL.MFE bit if multicast filtering is required.

Set up the VLAN Filter Table Array (VFTA) if VLAN support is required. This entire table should be zeroed and only the desired VLAN addresses should be permitted (by writing 0x1 to the corresponding bit location). Set the VLNCTRL.VFE bit if VLAN filtering is required.

Initialize the flexible filters 0…5 — Flexible Host Filter Table registers (FHFT).

After all memories in the filter units previously indicated are initialized, enable ECC reporting by setting the RXFECCERR0.ECCFLT_EN bit.

Program the different Rx filters and Rx offloads via registers FCTRL, VLNCTRL, MCSTCTRL, RXCSUM, RQTC, RFCTL, MPSAR, RSSRK, RETA, SAQF, DAQF, SDPQF, FTQF, SYNQF, ETQF, ETQS, RDRXCTL, RSCDBU.

Note that RDRXCTL.CRCStrip and HLREG0.RXCRCSTRP must be set to the same value. At the same time the RDRXCTL.RSCFRSTSIZE should be set to 0x0 as opposed to its hardware default.

Program RXPBSIZE, MRQC, PFQDE, RTRUP2TC, MFLCN.RPFCE, and MFLCN.RFCE according to the DCB and virtualization modes (see Section 4.6.11.3).

Enable jumbo reception by setting HLREG0.JUMBOEN in one of the following two cases:

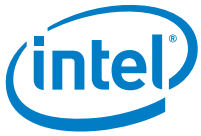
1. Jumbo packets are expected. Set the MAXFRS.MFS to expected max packet size.
2. LinkSec encapsulation is expected.

In these cases set the MAXFRS.MFS bit in the Max Frame Size register to the expected maximum packet size plus 32 bytes for the LinkSec encapsulation. Refer to Section 8.2.3.22.13 for details about the correct handling of VLAN and double VLAN headers.

Enable receive coalescing if required as described in Section 4.6.7.2.

The following should be done per each receive queue:

1. Allocate a region of memory for the receive descriptor list.
2. Receive buffers of appropriate size should be allocated and pointers to these buffers should be stored in the descriptor ring.
3. Program the descriptor base address with the address of the region (registers RDBAL, RDBAL).
4. Set the length register to the size of the descriptor ring (register RDLEN).
5. Program SRRCTL associated with this queue according to the size of the buffers and the required header control.
6. If header split is required for this queue, program the appropriate PSRTYPE for the appropriate headers.
7. Program RSC mode for the queue via the RSCCTL register.
8. Program RXDCTL with appropriate values including the queue *Enable* bit. Note that packets directed to a disabled queue are dropped.
9. Poll the RXDCTL register until the *Enable* bit is set. The tail should not be bumped before this bit was read as 1b.
10. Bump the tail pointer (RDT) to enable descriptors fetching by setting it to the ring length minus one.
11. Enable the receive path by setting RXCTRL.RXEN. This should be done only after all other settings are done following the steps below.
    - Halt the receive data path by setting SECRXCTRL.RX_DIS bit.
    - Wait for the data paths to be emptied by HW. Poll the SECRXSTAT.SECRX_RDY bit until it is asserted by HW.
    - Set RXCTRL.RXEN
    - Clear the SECRXCTRL.SECRX_DIS bits to enable receive data path



— If software uses the receive descriptor minimum threshold Interrupt, that value should be set.

Set bit 16 of the CTRL_EXT register and clear bit 12 of the DCA_RXCTRL[n] register[n].

4.6.7.1 Dynamic Enabling and Disabling of Receive Queues

Receive queues can be enabled or disabled dynamically using the following procedure.

4.6.7.1.1 Enabling

Follow the per queue initialization described in the previous section.

4.6.7.1.2 Disabling

- Disable the routing of packets to this queue by re-configuring the Rx filters. In order to ensure that the receive packet buffer does not contain any packets to the specific queue it is required to follow the Flushing the Packet Buffers procedure described later in this section.
- If RSC is enabled on the specific queue and VLAN strip is enabled as well then wait 2 ITR expiration time (ensure all open RSC are completed).
- Disable the queue by clearing the RXDCTL.ENABLE bit. The 82599 stops fetching and writing back descriptors from this queue. Any further packet that is directed to this queue is dropped. If a packet is being processed, the 82599 completes the current buffer write. If the packet spreads over more than one data buffer, all subsequent buffers are not written.
- The 82599 clears the RXDCTL.ENABLE bit only after all pending memory accesses to the descriptor ring are done. The driver should poll this bit before releasing the memory allocated to this queue.
- Once the RXDCTL.ENABLE bit is cleared the driver should wait additional amount of time
(~100 μs) before releasing the memory allocated to this queue.

Software might re-configure the Rx filters back to the original setting.The Rx path can be disabled only after all the receive queues are disabled.

4.6.7.1.3 Flushing the Packet Buffers

Because there could be additional packets in the receive packet buffer targeted to the disabled queue and the arbitration could be such that it would take a long time to drain these packets, if software re-enables a queue before all packets to that queue were drained, the enabled queue could potentially get packets directed to the old configuration of the queue. For example, a Virtual Machine (VM) goes down and a different VM gets the queue.

The 82599 provides a mechanism for software to identify when the packet buffers were drained of such stale packets. The read-only RXMEMWRAP register contains a set of counters (one per packet buffer) that increments each time a buffer is overtaken by the tail pointer. Software must read a counter repeatedly until its count is incremented at least by two, to insure that the buffer made at least one complete wrap-around. Software should also check the *Empty* bit for the counter. If the bit is set, the buffer is empty and there is no further need to sample the buffer counter.

4.6.7.2 RSC Enablement

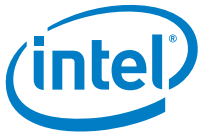
RSC enablement as well as RSC parameter settings are assumed as static. It should be enabled prior to reception and can be disabled only after the relevant Rx queue(s) are disabled.

4.6.7.2.1 Global Setting

- In SR-IOV mode RSC must be disabled globally by setting the RFCTL.RSC_DIS bit. In this case the following steps in this section are not required.
- Enable global CRC stripping via HLREG0 (hardware default setting)
- Software should set the RDRXCTL.RSCACKC bit that forces RSC completion on any change of the ACK bit in the Rx packet relative to the RSC context.
- The SRRCTL[n].BSIZEHEADER (header buffer size) must be larger than the packet header (even if header split is not enabled). A minimum size of 128 bytes for the header buffer addresses this requirement.
- NFS packet handling:
  - NFS header filtering should be disabled if NFS packets coalescing are required (at the TCP layer). The RFCTL.NFSW_DIS and RFCTL.NFSR_DIS bits should be set to 1b. Furthermore, the PSR_type1 bit in the PSRTYPE[n] registers (header split on NFS) must be turned off in all queues.
  - Both RFCTL.NFSW_DIS and RFCTL.NFSR_DIS bits should be cleared to 0b if NFS coalescing is not required. The PSR_type1 can be set per queue according to the required header split.

4.6.7.2.2 Per Queue Setting

- Enable RSC and configure the maximum allowed descriptors per RSC by setting the *MAXDESC* and *RSCEN* fields in the RSCCTL[n].
- Use non-legacy descriptor type by setting SRRCTL[n].DESCTYPE to non-zero values.
- TCP header recognition — the PSR_type4 in the PSRTYPE[n] registers should be set.
- Interrupt setting:
  - Interrupt moderation must be enabled by setting EITR[n].ITR_INTERVAL to a value greater than zero. Note that the ITR Interval must be larger than the RSC Delay. Also, if the CNT_WDIS bit is cleared (write enable), then the ITR counter should be set to zero.



— The RSC_DELAY field in the GPIE register should be set to the expected system latency descriptor write-back cycles. 4 to 8 µs should be sufficient in most cases. If software encounters many instances that RSC did not complete as expected following EITR interrupt assertion, *RSC Delay* might need to be increased.

— Map the relevant Rx queues to an interrupt by setting the relevant IVAR registers.

4.6.8 Transmit Initialization

- Program the HLREG0 register according to the required MAC behavior.
- Program TCP segmentation parameters via registers DMATXCTL (while maintaining *TE* bit cleared), DTXTCPFLGL, and DTXTCPFLGH; and DCA parameters via DCA_TXCTRL.
- Set RTTDCS.ARBDIS to 1b.
- Program DTXMXSZRQ, TXPBSIZE, TXPBTHRESH, MTQC, and MNGTXMAP, according to the DCB and virtualization modes (see Section 4.6.11.3).
- Clear RTTDCS.ARBDIS to 0b.

The following steps should be done once per transmit queue:

1. Allocate a region of memory for the transmit descriptor list.
2. Program the descriptor base address with the address of the region (TDBAL, TDBAH).
3. Set the length register to the size of the descriptor ring (TDLEN).
4. Program the TXDCTL register with the desired TX descriptor write back policy (see Section 8.2.3.9.10 for recommended values).
5. If needed, set TDWBAL/TWDBAH to enable head write back.
6. Enable transmit path by setting DMATXCTL.TE. This step should be executed only for the first enabled transmit queue and does not need to be repeated for any following queues.
7. Enable the queue using TXDCTL.ENABLE. Poll the TXDCTL register until the *Enable* bit is set.

Note: The tail register of the queue (TDT) should not be bumped until the queue is enabled.

4.6.8.1 Dynamic Enabling and Disabling of Transmit Queues

Transmit queues can be enabled or disabled dynamically if the following procedure is followed.

4.6.8.1.1 Enabling

Follow the per queue initialization described in the previous section.

4.6.8.1.2 Disabling

1. Stop storing packets for transmission in this queue.
2. The completion of the last transmit descriptor must be visible to software in order to guarantee that packets are not lost in step 5. Therefore, its *RS* bit must be set or WTHRESH must be greater than zero. If none of these conditions are met, software should add a null Tx data descriptor with an active *RS* bit.
3. Wait until the software head of the queue (TDH) equals the software tail (TDT), indicating the queue is empty.
4. Wait until all descriptors are written back (polling *DD* bit in ring or polling the Head_WB content). It might be required to flush the transmit queue by setting TXDCTL[n].SWFLSH if the *RS* bit in the last fetched descriptor is not set or if WTHRESH is greater than zero.
5. Disable the queue by clearing TXDCTL.ENABLE.
6. Any packets waiting for transmission in the packet buffer would still be sent at a later time.

The transmit path can be disabled only after all transmit queues are disabled.

4.6.9 FCoE Initialization Flow

Ordering between the following steps is not critical as long as it is done before transmit and receive starts.

- The FCoE DDP context table should be initialized clearing the FCBUFF.Valid and FCFLT.Valid bits of all contexts.
- EType Queue Filter — ETQF[n]: Select a filter by setting the FCoE bit. The *EType* field should be set to 0x8906 (FCoE Ethernet Type). If FCoE traffic is expected on multiple VLAN priorities then multiple ETQF filters might be required.
- EType Queue Select — ETQS[n]: Each ETQF filter is associated to a queue select register. The ETQS registers can be used to direct the FCoE traffic to specific receive queues. Up to one queue per Traffic Class (TC) as programmed in the ETQF.
- Multiple receive queues can be enabled by setting FCRECTL.ENA and programming the FCRETA[n] registers.
- Low Latency Interrupts (LLI) for critical FCoE frames can be enabled by setting the FCRXCTRL.FCOELLI bit.
- Set the RDRXCTL.FCOE_WRFIX bit that forces a DDP write exchange context closure after receiving the last packet in a sequence with an active *Sequence Initiative* bit in the *F_CTL* field.
- Follow the rules indicated in Section 7.13.2.1 and Section 7.13.3.1 for Tx and Rx cross functionality requirements. These sections include requirements on Ethernet CRC and padding handling, LinkSec offload, Legacy Rx buffers, and more.
- Software is not expected to access the EOF and SOF setting. If there is some error in the implementation, these settings might need to be modified. If so, software must program the REOFF and TEOFF registers by the same values. This same rule applies to the RSOFF and TSOFF registers.

4.6.10 Virtualization Initialization Flow

4.6.10.1 VMDq Mode

4.6.10.1.1 Global Filtering and Offload Capabilities

- Select one of the VMDQ pooling methods — MAC/VLAN filtering for pool selection and either DCB or RSS for the queue in pool selection. MRQC.Multiple Receive Queues Enable = 1000b, 1010b, 1011b, 1100b, or 1101b.
- DCB should be initiated as described in Section 4.6.11. In RSS mode, the RSS key (RSSRK) and redirection table (RETA) should be programmed. Note that the redirection table is common to all the pools and only indicates the queue inside the pool to use once the pool is chosen. Each pool can decide if it uses DCB.
- Configure PFVTCTL to define the default pool.
- Enable replication via PFVTCTL.Rpl_En.
- If needed, enable padding of small packets via HLREG0.TXPADEN.
- The MPSAR registers are used to associate Ethernet MAC addresses to pools. Using the MPSAR registers, software must reprogram RAL[0] and RAH[0] by their values (software could read these registers and then write them back with the same content).

4.6.10.1.2 Mirroring Rules

For each mirroring rule to be activated:

- Set the type of traffic to be mirrored in the PFMRCTL[n] register.
- Set the mirror pool in PFMRCTL[n].MP.
- For pool mirroring, set the PFMRVM[n] register with the pools to be mirrored.
- For VLAN mirroring, set PFMRVLAN[n] with the indexes from the PFVLVF registers of the VLANs to be mirrored.

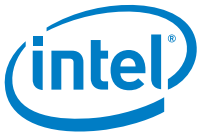
4.6.10.1.3 Security Features

For each pool, the driver might activate the MAC and VLAN anti-spoof features via the relevant bit in PFVFSPOOF.MACAS and PFVFSPOOF.VLANAS, respectively.

4.6.10.1.4 Per Pool Settings

As soon as a pool of queues is associated to a VM, software should set the following parameters:

- Associate the unicast Ethernet MAC address of the VM by enabling the pool in the MPSAR registers.
- If all the Ethernet MAC addresses are used, the Unicast Hash Table (PFUTA) can be used. Pools servicing VMs whose address is in the hash table should be declared as so by setting PFVML2FLT.ROPE. Packets received according to this method didn't pass perfect filtering and are indicated as such.
- Enable the pool in all the RAH/RAL registers representing the multicast Ethernet MAC addresses this VM belongs to.
- If all the Ethernet MAC addresses are used, the Multicast Hash Table (MTA) can be used. Pools servicing VMs using multicast addresses in the hash table should be declared as so by setting PFVML2FLT.ROMPE. Packets received according to this method didn't pass perfect filtering and are indicated as such.
- Define whether this VM should get all multicast/broadcast packets in the same VLAN via PFVML2FLT.MPE and PFVML2FLT.BAM, and whether it should accept untagged packets via PFVML2FLT.AUPE.
- Enable the pool in each PFVLVF and PFVLVFB registers this VM belongs to.
- A VM might be set to receive it's own traffic in case the source and the destination are in the same pool via the PFVMTXSW.LLE.
- Whether VLAN header and CRC should be stripped from the packet. Note that even if the CRC is kept, it might not match the actual content of the forwarded packet, because of other offloads application such as VLAN stripor LinkSec decrypting.
- Set which header split is required via the PSRTYPE register.
- In RSS mode, define if the pool uses RSS via the proper MRQC.MRQE mode.
- Enable the Pool in the PFVFRE register to allow Rx Filtering
- To Enable Multiple Tx queues, Set the MTQC as described in Section 7.2.1.2.1
- Enable the Pool in the PFVFTE register to allow Tx Filtering
- Enable Rx and Tx queues as described in Section 4.6.7 and Section 4.6.8.
- For each Rx queue a drop/no drop flag can be set in SRRCTL.DROP_EN and via the PFQDE register, controlling the behavior if no receive buffers are available in the queue to receive packets. The usual behavior is to allow drop in order to avoid head of line blocking. Setting PFQDE per queue is made by using the *Queue Index* field in the PFQDE register.



4.6.10.2 IOV Initialization

4.6.10.2.1 Physical Function (PF) Driver Initialization

The PF driver is responsible for the link setup and handling of all the filtering and offload capabilities for all the VFs as described in Section 4.6.10.1.1 and the security features as described in Section 4.6.10.1.3. It should also set the bandwidth allocation per transmit queue for each VF as described in Section 4.6.10.

Note: The link setup might include the authentication process (802.1X or other), and setup of the LinkSec channel, and setup of the DCB parameters.

In IOV mode, VMDq + RSS mode is not available.

After all the common parameters are set, the PF driver should set all the VFMailbox[n].RSTD bits by setting CTRL_EXT.PFRSTD.

PF enables VF traffic via the PFVFTE and PFVFRE registers after all VF parameters are set as defined in Section 4.6.10.1.4.

Note: If the operating system changes the NumVF setting in the PCIe SR-IOV Num VFs register after the device was active, it is required to initiate a PF software reset following this change.

4.6.10.2.1.1 VF Specific Reset Coordination

After the PF driver receives an indication of a VF FLR via the PFVFLRE register, it should enable the receive and transmit for the VF only once the device is programmed with the right parameters as defined in Section 4.6.10.1.4. The receive filtering is enabled using the PFVFRE register and the transmit filtering is enabled via the PFVFTE register.

Note: The filtering and offloads setup might be based on central IT settings or on requests from the VF drivers.

4.6.10.2.2 VF Driver Initialization

Upon initialization, after the PF indicated that the global initialization was done via the VFMailbox.RSTD bit, the VF driver should communicate with the PF, either via the mailbox or via other software mechanisms to assure that the right parameters of the VF are programmed as described in Section 4.6.10.1.4. The PF driver might then send an acknowledge message with the actual setup done according to the VF request and the IT policy.

The VF driver should then setup the interrupts and the queues as described in Section 4.6.6, Section 4.6.7 and Section 4.6.8.

4.6.10.2.3 Full Reset Coordination

A mechanism is provided to synchronize reset procedures between the PF and the VFs. It is provided specifically for PF software reset but can be used in other reset cases. These reset cases are described in the following procedure.

One of the following reset cases takes place:

- LAN Power Good

- PCIe reset (PERST and in-band)
- D3hot --> D0
- FLR
- Software reset by the PF

The 82599 sets the *RSTI* bits in all the VFMailbox registers. Once the reset completes, each VF might read its VFMailbox register to identify a reset in progress.

Once the PF completes configuring the device, it clears the CTRL_EXT.PFRSTD bit. As a result, the 82599 clears the *RSTI* bits in all the VFMailbox registers and sets the *RSTD* (*Reset Done*) bits in all the VFMailbox registers.

Until a RSTD condition is detected, the VFs should access only the VFMailbox register and should not attempt to activate the interrupt mechanism or the transmit and receive process.

4.6.11 DCB Configuration

After power up or device reset, DCB and any type of FC are disabled by default, and a unique TC and packet buffer (like PB0) is used. In this mode, the host can exchange information via DCX protocol to determine the number of TCs to be configured. Before setting the device to multiple TCs, it should be reset (software reset).

The registers concerned with setting the number of TCs are: RXPBSIZE[0-7], TXPBSIZE[0-7], TXPBTHRESH, MRQC, MTQC, and RTRUP2TC registers along with the following bits RTRPCS.RAC, RTTDCS.TDPAC, RTTDCS.VMPAC and RTTPCS.TPPAC.

They cannot be modified on the fly, but only after device reset. Packet buffers with a non-null size must be allocated from PB0 and up.

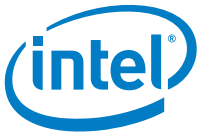
Rate parameters and bandwidth allocation to VMs can be modified on the fly without disturbing traffic flows.

4.6.11.1 CPU Latency Considerations

When the CPU detects an idle period of some length, it enters a low-power sleep state. When traffic arrives from the network, it takes time for the CPU to wake and respond (such as to snoop). During that period, Rx packets are not posted to system memory.

If the entry time to sleep state is too short, the CPU might be getting in and out of sleep state in between packets, therefore impacting latency and throughput. 100 µs was defined as a safe margin for entry time to avoid such effects.

Each time the CPU is in low power, received packets need to be stored (or dropped) in the 82599 for the duration of the exit time. Given 64 KB Rx packet buffers per TC in the 82599, Priority Flow Control (PFC) does not spread (or a packet is not dropped) provided that the CPU exits its low power state within
50 µs.



4.6.11.2 Link Speed Change Procedure

Each time the link status or speed is changed, hardware is automatically updating the transmit rates that were loaded by software relatively to the new link speed. This means that if a rate limiter was set by software to 500 Mb/s for a 10 GbE link speed, it is changed by hardware to 50 Mb/s if the link speed has changed to 1 GbE.

Since transmit rates must be considered as absolute rate limitations (expressed in Mb/s, regardless of the link speed), in such occasions software is responsible to either clear all the transmit rate-limiters via the BCN_CLEAR_ALL bit in RTTBCNRD register, and/or to re-load each transmit rate with the correct value relatively to the new link speed. In the previous example, the new transmit rate value to be loaded by software must be multiplied by 10 to maintain the rate limitation to 500 Mb/s.

4.6.11.3 Initial Configuration Flow

Only the following configuration modes are allowed.

4.6.11.3.1 General Case: DCB-on, VT-on

1. Configure packet buffers, queues, and traffic mapping:
   - 8 TCs mode — Packet buffer size and threshold, typically

     RXPBSIZE[0-7].SIZE=0x40

     TXPBSIZE[0-7].SIZE=0x14

     but non-symmetrical sizing is also allowed (see Section 8.2.3.9.13 for rules)

     TXPBTHRESH.THRESH[0-7]=TXPBSIZE[0-7].SIZE — Maximum expected Tx packet length in this TC
   - 4 TCs mode — Packet buffer size and threshold, typically

     RXPBSIZE[0-3].SIZE=0x80, RXPBSIZE[[4-7].SIZE=0x0

     TXPBSIZE[0-3].SIZE=0x28, TXPBSIZE[4-7].SIZE=0x0

     but non-symmetrical sizing among TCs[0-3] is also allowed (see Section 8.2.3.9.13 for rules)

     TXPBTHRESH.THRESH[0-3]=TXPBSIZE[0-3].SIZE — Maximum expected Tx packet length in this TC

     TXPBTHRESH.THRESH[4-7]=0x0
   - Multiple Receive and Transmit Queue Control (MRQC and MTQC)
     - Set MRQC.MRQE to 1xxxb, with the three least significant bits set according to the number of VFs, TCs, and RSS mode as described in Section 8.2.3.7.12.
     - Set both *RT_Ena* and *VT_Ena* bits in the MTQC register.
     - Set MTQC.NUM_TC_OR_Q according to the number of TCs/VFs enabled as described in Section 8.2.3.7.16.
   - Set the PFVTCTL.VT_Ena (as the MTQC.VT_Ena)
   - Queue Drop Enable (PFQDE) — In SR-IO the *QDE* bit should be set to 1b in the PFQDE register for all queues. In VMDq mode, the *QDE* bit should be set to 0b for all queues.

— Split receive control (SRRCTL[0-127]): Drop_En=1 — drop policy for all the queues, in order to avoid crosstalk between VMs

— Rx User Priority (UP) to TC (RTRUP2TC)

— Tx UP to TC (RTTUP2TC)

— DMA TX TCP Maximum Allowed Size Requests (DTXMXSZRQ) — set Max_byte_num_req = 0x010 = 4 KB

2. Enable PFC and disable legacy flow control:

— Enable transmit PFC via: FCCFG.TFCE=10b

— Enable receive PFC via: MFLCN.RPFCE=1b

— Disable receive legacy flow control via: MFLCN.RFCE=0b

— Refer to Section 8.2.3.3 for other registers related to flow control

3. Configure arbiters, per TC[0-1]:

— Tx descriptor plane T1 Config (RTTDT1C) per queue, via setting RTTDQSEL first. Note that the RTTDT1C for queue zero must always be initialized.

— Tx descriptor plane T2 Config (RTTDT2C[0-7])

— Tx packet plane T2 Config (RTTPT2C[0-7])

— Rx packet plane T4 Config (RTRPT4C[0-7])

4. Enable TC and VM arbitration layers:

— Tx Descriptor plane Control and Status (RTTDCS), bits:

TDPAC=1b, VMPAC=1b, TDRM=1b, BDPM=1b, BPBFSM=0b

— Tx Packet Plane Control and Status (RTTPCS): TPPAC=1b, TPRM=1b, ARBD=0x004

— Rx Packet Plane Control and Status (RTRPCS): RAC=1b, RRM=1b

5. Set the SECTXMINIFG.SECTXDCB field to 0x1F.

4.6.11.3.2 DCB-On, VT-Off

Set the configuration bits as specified in Section 4.6.11.3.1 with the following exceptions:

- Configure packet buffers, queues, and traffic mapping:

— MRQC and MTQC

  - Set MRQE to 0xxxb, with the three least significant bits set according to the number of TCs and RSS mode
  - Set *RT_Ena* bit and clear the *VT_Ena* bit in the MTQC register.
  - Set MTQC.NUM_TC_OR_Q according to the number of TCs enabled

— Clear PFVTCTL.VT_Ena (as the MRQC.VT_Ena)

- Allow no-drop policy in Rx:

— PFQDE: The *QDE* bit should be set to 0b in the PFQDE register for all queues enabling per queue policy by the SRRCTL[n] setting.