VMS Software

VSI LAN Devices, Counters, and Functions Reference Manual for OpenVMS x86-64

Operating System and Version: VSI OpenVMS x86-64 Version 9.2-3 or higher

VMS Software, Inc. (VSI)
Boston, Massachusetts, USA

VSI LAN Devices, Counters, and Functions Reference Manual for OpenVMS x86-64

Legal Notice

Table of Contents

Chapter 1. Introduction

This document describes the driver-specific counters and functions maintained by the LAN drivers internally on VSI OpenVMS x86-64 systems.

The LAN drivers maintain the following information that the **LAN Control Program** (LANCP) and the LAN **System Dump Analyzer** (SDA) extension can display:

- Device counters, packets, and bytes sent and received and error counters for transmit and receive.
- Driver-specific internal counters, device and driver settings, and counters maintained internal to the driver. This includes driver messages showing settings and link state changes.
- Error log information.
- Driver tracing (by default, mostly errors), state changes, and link events. Selecting which events to trace is done using LANCP commands and LAN_FLAGS system parameter settings.
- Bandwidth monitoring. Adjusting bandwidth monitoring settings is done using LANCP commands and LAN_FLAGS settings.

The LAN drivers provide additional capabilities, mostly for diagnostic and test purposes. These capabilities are called **device-specific functions** and are initiated by LANCP commands. They can be used to reset a device, cause various errors, set promiscuous mode, and so on. The selection of functions is driver specific.

The devices covered in this document are as follows:

- Intel 1Gb/emulated Ethernet
- KVM VirtIO Ethernet
- VMware VMXNET3 Ethernet

Chapter 2. Intel 1Gb Ethernet

The Intel 1Gb devices supported for OpenVMS x86-64 are as follows:

Vendor ID	Sub IDs	Bus	Device Name	Chip
100E8086	001E8086	PCI	e1000	82540VM (VBOX)
10048086	catchall	PCI	e1000	82543VM (VBOX)
100F8086	catchall	PCI/PCIe	e1000	82545VM (VBOX, VMware)
100E8086	11001AF4	PCIe	e1000	82540VM (KVM)
10D38086	catchall	PCI/PCIe	e1000e	82574VM (KVM, VMware)

The driver is SYS$EI1000X.EXE.

There are multiple names for each device, as follows:

- PCI ID repository.
- Vendor names and part numbers (there may be multiple names and part numbers).
- Full name assigned by the driver, which includes a part number if known.

  The full name is specified by the "device =" parameter in the entry in SYS$SYSTEM:SYS$CONFIG.DAT and is the same name assigned by the driver. The full name can be found in the list of devices displayed by the **`SEARCH SYS$SYSTEM:SYS$CONFIG.DAT DEVICE,"="/MATCH=AND`** command.

  The full name is also displayed by the LANCP **`SHOW DEVICE/INTERNAL`** command and the SDA **`CLUE CONFIG/ADAPTER`** command.
- Short name assigned by the driver that is used for output requiring a shorter name format.

  The short name is displayed by the LANCP **`SHOW CONFIG`** and **`SHOW DEVICE/INTERNAL`** commands.
- Additional names or aliases are given in the comments before the SYS$SYSTEM:SYS$CONFIG.DAT entry. For example, this device:

  ```
  HPE 4-port Intel I350 366FLR Adapter (665240-B21) (Gigabit Ethernet)
  ```

  is specified in SYS$SYSTEM:SYS$CONFIG.DAT, followed by comments that give other names that identify the NIC:

  ```
  ! HP Ethernet 1GB 4-port 366FLR Adapter (665240-B21)             I350
  ! HPE Ethernet 1GB 4-port FLR-T I350-T4V2 (665240-B21)           I350
  ! HPE 4-port FLR-T Intel I350-T4V2 (665240-B21)                  I350
  device = "HPE 4-port Intel I350 366FLR Adapter (665240-B21) (Gigabit Ethernet)"
  ```

The PCI Device IDs are stored in the bus array entry. To find the IDs for a particular device, execute the SDA **`CLUE CONFIG/ADAPTER`** and **`EXAMINE busarray-address`** commands; the Device ID, Vendor ID, SubDevice ID, and SubVendor ID are the first two longwords.

2.1. Intel 1Gb Internal Counters

The LANCP command **SHOW DEVICE/INTERNAL_COUNTERS EIA** displays the entire set of internal counters maintained by the Intel 1Gb driver. Some counters are special debug counters and are not displayed unless the additional qualifier **/DEBUG** is specified. Counters that are zero are not displayed unless the additional qualifier **/ZERO** is specified.

The LAN SDA extension also displays the complete set of internal counters with the **LAN INTERNAL/DEVICE=EIA** command.

Note

The definitions of the following counters may change between versions of the driver.

Counters

Device name (full)
Device name (short)

Name of the Intel 1Gb device.

Driver timestamp

Compilation timestamp of the driver.

Driver version

Driver version numbered 1–*n* that is usually identical to the *x*-*n* ID displayed in the **ANALYZE/IMAGE** command output for the driver image. It includes variant information, if applicable. The full driver version includes the target OpenVMS release and is displayed by the SDA **LAN/DEVICE=EIA** command in the quadword driver version field.

Device revision

Hardware revision level of the chip.

Device interrupts

Number of times the driver interrupt service routine was called.

Link transitions

Number of link transitions seen by the driver.

Interrupt link state changes

Number of link state changes reported in the interrupt control register.

Link checks

Number of times the driver checked the current link status.

Link setup delay (µs)

Total time (in microseconds) waiting for the device during setup of the link.

Maximum link setup delay (µs)

Maximum time (in microseconds) waiting for the device during setup of the link.

VCI port usable/unusable events reported

Number of VMS Communications Interface (VCI) user port events reported.

Device resets

Number of times the device was reset.

CSR Base Address

PCI BAR0 Control and Status Register (CSR) base address.

Unit inits

Number of unit initializations executed; since unit initialization is only executed once, this counter will be 1.

User start/change/stop requests

Number of user startup and shutdown requests processed by the driver; this is typically one or two requests when a user starts up and one when a user stops.

Transmits queued

Number of transmit requests queued because the link was not available or because too many transmit requests were already outstanding.

Transmit timeouts

Number of times the driver has timed out a transmit, reset the device, and completed outstanding I/O with error status.

Transmit timeouts (averted)

Number of transmit timeouts averted by a final check for transmit completion.

Transmit chained (buffer address)

Number of chained transmits where the chaining is via system virtual address (SVA) buffer address.

Transmit chained (SVAPTE_SVA)

Number of chained transmits where the chaining is via a SVAPTE_SVA-style buffer.

Transmit chained (IOBD/Extent)

Number of chained transmits where the chaining is via an I/O Buffer Descriptor (IOBD)/Extent-style buffer.

Transmit errors (too few segments)

Number of transmit requests completed with error status (SS$_INCSEGTRA) because the application did not specify the transmit buffer completely.

Transmit informationals (zero-byte segments)

Number of transmit chain segments which specified a zero-byte length segment.

Transmit copies (too many segments)

Number of transmit requests that exceeded the maximum number of chain segments the driver can handle; the driver then copied some of them to a temporary buffer so that it could transmit the packet.

Transmit copy failures

Number of transmit failures caused by too few chain segments when coalescing a request to reduce the number of segments needed.

Jumbo transmits issued

Number of transmit requests with a packet length exceeding 1514 bytes (excluding CRC).

Jumbo receives issued

Number of jumbo receive buffers allocated and given to the device.

Chained receives completed

Number of receives (jumbo packets) completed by the driver that span multiple receive buffers.

Receives discarded (bad frame)

Number of receive packets discarded by the driver due to a receive error. By default, the chip does not deliver damaged receive packets to the driver, but it can be instructed to do so with a device-specific function.

Soft errors

Number of times errors were recovered in the driver by resetting and reinitializing the device without notifying user applications.

Rescheduled forks (too long in fork)

Number of times that a rescheduled fork was done. In transmit and receive processing, the driver limits the amount of time spent in the fork process before rescheduling.

PHY register read/write errors

Number of errors reading or writing a Physical Layer (PHY) register.

Standard packet size (bytes)

Ethernet packet size (1518 bytes).

Jumbo packet size (bytes)

Jumbo packet size (9018 bytes).

Requested link state

Speed and duplex mode requested by a user.

Current link state

Current link state.

Driver flags

Driver flags.

Driver state

Current driver state, as follows:

0 – Driver state is undefined.
1 – Driver is initializing the device.
2 – Driver is running, but the link is down. Completing transmits with error status.
4 – Driver is running, but the link is down. Queuing transmits for now.
8 – Driver is running, and the link is up. Processing transmits and receives.
16 – Device is not usable.

LAN_FLAGS system parameter

Current value of the LAN_FLAGS system parameter.

Packet buffer allocation (initial value)

Amount of the chip packet buffer allocated to transmit buffers vs receive, initial value.

Packet buffer allocation (current value)

Amount of the chip packet buffer allocated to transmit buffers vs receive, current value.

Interrupt delay value

Minimum delay between interrupts, in units of 256 nanoseconds.

Transmit interrupt delay (µs)

Amount of delay (in microseconds) after transmit completion that an interrupt is generated.

Receive interrupt delay (µs)

Amount of delay (in microseconds) after receive completion that an interrupt is generated.

Receive rings VA

System virtual address (VA) of the start of the receive rings.

Transmit rings VA

System VA of the start of the transmit rings.

LSB size

Total size of the LAN Station Block (LSB) structure.

Interrupt mode

Unknown, IOSAPIC, MSI, MSI-X, or IOAPIC.

Transmit time limit

Transmit time limit in seconds after which a timeout is declared.

Timer routine interval

Resolution of the transmit timer (the real timeout is limit + interval).

Time Stamps

Current uptime

Current system uptime.

Last reset

Last time the device was reset.

Last link state change interrupt

Last time a link state change interrupt was fielded.

Previous link state change interrupt

Previous time a link state change interrupt was fielded.

Last link up

Last time a link up transition occurred.

Last link down

Last time a link down transition occurred.

Total link uptime

Total time the link has been up.

Total link downtime

Total time the link has been down.

Time of last uplink period
Time of last uplink period-1
Time of last uplink period-2
Time of last uplink period-3
Time of last uplink period-4
Time of last uplink period-5
Time of last uplink period-6
Time of last uplink period-7
Time of last uplink period-8
Time of last uplink period-9
Time of last uplink period-10
Time of last uplink period-11
Time of last uplink period-12
Time of last uplink period-13
Time of last uplink period-14

History of the last 15 link uptimes, displaying the length of each uptime period.

Last transmit timeout

Time of the last transmit timeout.

Last transmit timeout (averted)

Time last averted a transmit timeout.

Last soft error

Time of the last soft error.

Last CRC error

Time of the last Cyclic Redundancy Check (CRC) error.

Last receive error

Time of the last receive error.

Last receive error packet header

First 24 bytes of the last packet received with error status.

Last unrecognized unicast packet

Time of the last unrecognized unicast packet received.

Last unrecognized unicast packet header

First 24 bytes of the last unicast packet received that was discarded because receive address filtering did not result in a matching user.

Last unrecognized multicast packet

Time of the last unrecognized multicast packet received.

Last unrecognized multicast packet header

First 24 bytes of the last multicast packet received that was discarded because receive address filtering did not result in a matching user.

Device Registers (read/wrote)

Consists of a list of the contents-significant chip registers.

PCI Config Registers

- Configuration ID (CFID) register
- Configuration Command Register (CFCR)
- Configuration Revision (CFRV) register
- Configuration Latency Timer (CFLT) register
- PCI_ADDRESS0 register

- PCI_ADDRESS1 register
- SUB_VNDR / SUB_ID register
- CACHE_LINE_SIZE register

Receive Buffers

Minimum buffers requested

Minimum number of receive buffers as set by default or by management request (SETMODE or LANCP command). This is used to determine the "Current minimum limit".

Maximum buffers requested

Maximum number of receive buffers as set by default or by management request (SETMODE or LANCP command). This is used to determine the "Current maximum limit".

Current minimum limit

Minimum number of receive buffers as determined by the driver and by management request, "Minimum buffers requested". The driver will not allow the number of receive buffers in its receive queues to drop below this value. This does not include receive buffers transferred to applications (and not yet returned).

Current maximum limit

Maximum number of receive buffers as determined by the driver and by management request, "Maximum buffers requested". The driver will deallocate receive buffers until the number allocated does not exceed this maximum. When applications return receive buffers, the number may exceed this maximum, at which point they are deallocated. Note that the driver allows this maximum to be exceeded slightly to limit allocation and deallocation activity. See "Target number of buffers maximum" below.

Current number of buffers

Current number of receive buffers owned by the driver.

Target number of buffers

Desired number of receive buffers. As receive activity does not result in lost packets, this number is gradually decreased toward the "Current minimum limit". If packets are lost, this number is dramatically increased toward the "Current maximum limit". When the driver allocates receive buffers, it allocates them until reaching this target number.

Target number of buffers maximum

Maximum desired number of receive buffers. When an application returns a buffer to the driver, if the current number does not exceed this target maximum, the buffer is kept by the driver. Otherwise, it is deallocated.

Fork Delay (after scheduled)

To help determine whether the buffering requirements of the driver and the device are sufficient for the system configuration, the driver records the amount of time from fork scheduled to the time the fork is actually run. The data is recorded in 10-millisecond increments from 10 to 310 milliseconds.

This data can be used in conjunction with the number of packets discarded due to insufficient buffers to determine whether the buffering settings of the driver (minimum and maximum receive buffers) and the amount of buffering on the device are sufficient for normal operation. If packets are being discarded, the buffering should be increased until the number of packets lost is minimal.

Transmit Time

To help understand the operation of the system, driver, and device, the driver records the time that each transmit buffer is given to the device. When the transmit completes, the driver calculates the elapsed time, creating a histogram of transmit times from 10 to 310 milliseconds.

Receive completion time

To help understand the operation of the system and driver, the driver records the time taken to process each receive buffer. When the receive completes, the driver records the time that processing started. Then it delivers the packet to the user application. When the user returns it, the driver calculates this completion time, creating a histogram of receive times from 10 to 150 milliseconds.

One second timer time

The driver maintains a one-second timer (that runs four times a second). It expects the timer routine to be called close to the timer interval. By looking at the time difference between when a timer is expected to be called and when it actually is called, it is possible to understand the operation of the system and driver. The variance time is used to create a histogram of receive times from 10 to 150 milliseconds.

Statistics Block

These are the statistics kept by the device.

Driver Messages

Console messages issued by the driver.

2.2. Intel 1Gb Device-Specific Functions

The **`SET DEVICE /DEVICE_SPECIFIC=(FUNCTION="FUNC",VALUE=(V1,V2,...))`** LANCP command provides a mechanism to issue device-specific functions to the Intel 1Gb driver. While some functions are common to many LAN drivers, most are primarily useful in a diagnostic context and are not typically included in LANCP command documentation.

For further information on the LANCP utility and LANCP commands, refer to the relevant section in the *VSI OpenVMS System Management Utilities Reference Manual, Volume I: A–L* [https://docs.vmssoftware.com/vsi-openvms-system-management-utilities-reference-manual-volume-i-a-l/#d0e28590] or the LANCP online Help.

Notes

The function name is always four characters in length.

The command requires the SYSPRV privilege.

The definitions of the following functions may change between versions of the driver.

Device-Specific Functions

DEVICE_SPECIFIC=FUNCTION=("RCOR",VALUE=%X*ptyvalue*)
DEVICE_SPECIFIC=FUNCTION=("XCOR",VALUE=%X*ptyvalue*)

Corrupt a transmit or receive packet. When this function is executed, the protocol type is specified; if it matches on a transmit or receive packet, the next packet seen is corrupted by changing the last bit in the VMS Communications Request Packet (VCRP) or Complex Chained Buffer (CXB), XORing it with a 1. The protocol type is then cleared so that the corruption is not done again.

The following example shows how to corrupt a PEDRIVER transmit packet:

```
$ MCR LANCP SET DEVICE/TRACE=(MASK=XMTISSUE) EIA
$ MCR LANCP SET DEVICE/DEVICE_SPECIFIC=(FUNCTION="XCOR", VALUE=%x760) EIA
```

The following example shows how to corrupt a PEDRIVER receive packet:

```
$ MCR LANCP SET DEVICE/TRACE=(MASK=RCVDONE) EIA
$ MCR LANCP SET DEVICE/DEVICE_SPECIFIC=(FUNCTION="RCOR", VALUE=%x760) EIA
```

DEVICE_SPECIFIC=FUNCTION=("LOSE",VALUE=%x*aabbccdd*)

Lose some packets by corrupting the header of the packet. Corrupt packet 0x*aabbccdd*, where *aa* is seconds to corrupt, *bb* is number to corrupt, *cc* is byte number 0–255, and *dd* is what to corrupt with.

For example, to send 32 transmit packets elsewhere (no more than 16 seconds), change the last byte of the destination address to 0x44. For example:

```
$ MCR LANCP SET DEVICE/TRACE=(MASK=XMTISSUE) EIA
$ MCR LANCP SET DEVICE/DEVICE_SPECIFIC=(FUNCTION="LOSE", VALUE=%x10200544) EIA
```

To lose just one packet, change the first byte of the packet to 0x00. For example:

```
$ MCR LANCP SET DEVICE/TRACE=(MASK=XMTISSUE) EIA
$ MCR LANCP SET DEVICE/DEVICE_SPECIFIC=FUNCTION="LOSE" EIA
```

To lose 255 packets for the rest of the 10-millisecond tick, change the first byte of the packet to 0x00. For example:

```
$ MCR LANCP SET DEVICE/TRACE=(MASK=XMTISSUE) EIA
$ MCR LANCP SET DEVICE/DEVICE_SPECIFIC=(FUNCTION="LOSE",VALUE=%xFF0000) EIA
```

DEVICE_SPECIFIC=FUNCTION=("FCOP",VALUE=*setting*)

Force transmit chaining coalescing to exercise this rarely-used code path.

Enable (1) or disable (0) this function.

DEVICE_SPECIFIC=FUNCTION="DMIG"

Disable interrupt mitigation. Note that the Transmit Interrupt Delay Value (TIDV) register is set to the minimum value of 1 (specifications state that it cannot be set to 0).

This is used for performance testing when it is desirable to get an interrupt immediately upon transmit and receive completion.

DEVICE_SPECIFIC=(FUNCTION="DXMT",VALUE=*delayvalue*)

Change the interrupt mitigation delay on transmit completion value, the "Transmit interrupt delay (µs)" value. This value is in microseconds, limited to 1000. If zero, the delay is disabled.

DEVICE_SPECIFIC=(FUNCTION="DRCV",VALUE=*delayvalue*)

Change the interrupt mitigation delay on receive completion value, the "Receive interrupt delay (µs)" value. This value is in microseconds, limited to 1000. If zero, the delay is disabled.

DEVICE_SPECIFIC=(FUNCTION="DINT",VALUE=*delayvalue*)

Change the interrupt delay value. The value is in microseconds, limited to 1000. If zero, the delay is disabled.

DEVICE_SPECIFIC=(FUNCTION="FLOW",VALUE=*setting*)

Disable flow control (if the value is zero) or enable flow control (if the value is nonzero).

DEVICE_SPECIFIC=FUNCTION=("PROM",VALUE=*setting*)

Enable (1) or disable (0) force promiscuous mode, then reset the device.

DEVICE_SPECIFIC=FUNCTION=("APRM",VALUE=*setting*)

Enable (1) or disable (0) auto-promiscuous mode, then reset the device.

Auto-promiscuous mode means the driver will enable promiscuous mode if it needs to according to the hypervisor in use. Enabling promiscuous mode may be needed if the Media Access Control (MAC) address is changed from the default power on address.

DEVICE_SPECIFIC=FUNCTION="SOFT"

Force a non-fatal (soft) error. The device is restarted.

DEVICE_SPECIFIC=FUNCTION="HBUG"

Force a simulated hardware failure by clearing the receive return descriptor ring.

DEVICE_SPECIFIC=(FUNCTION="PRES",VALUE=*periodicmask*)

Schedule periodic random resets. The value supplied is AND'ed with (RSCC × 69 069) and, if the result is 3, a reset is done. For example, if the value supplied is 7, a reset is done every 8 seconds on average.

A value of zero disables this function, and a nonzero value enables it.

This is used to test error handling.

DEVICE_SPECIFIC=FUNCTION="FCRC"

Send the next packet without a valid CRC.

This is used to generate CRC errors for test purposes.

DEVICE_SPECIFIC=(FUNCTION="RBAD",VALUE=*value*)

If the value is zero, the chip will discard bad received packets. If the value is nonzero, the bad packets will be received and discarded by the driver. This allows the packets to be traced and inspected.

DEVICE_SPECIFIC=FUNCTION=("WCSR",VALUE=(*address*,*val*)).

Write a value *val* to CSR *address*.

DEVICE_SPECIFIC=FUNCTION=("RCSR",VALUE=*address*).

Read a CSR *address* and print the result.

DEVICE_SPECIFIC=FUNCTION=("WPHY",VALUE=(*address*,*val*)).

Write a value *val* to PHY *address*.

DEVICE_SPECIFIC=FUNCTION=("ALLM",VALUE=*setting*)

Enable (1) or disable (0) all messages flag.

DEVICE_SPECIFIC=FUNCTION=("FLIN",VALUE=*setting*)

Force link indication. If the value is 0 (disabled), do not force the link state.

If the value is 1, force the link to be up.

If the value is 2, force the link to be down.

DEVICE_SPECIFIC=FUNCTION=("XTOB",VALUE=*setting*)

Enable (1) or disable (0) Bugcheck on transmit timeout.

DEVICE_SPECIFIC=FUNCTION=("XLON",VALUE=*setting*)

Enable (1) or disable (0) Bugcheck on long transmit completion.

DEVICE_SPECIFIC=FUNCTION=("FLON",VALUE=*setting*)

Enable (1) or disable (0) Bugcheck on long fork.

DEVICE_SPECIFIC=FUNCTION=("SLON",VALUE=*setting*)

Enable (1) or disable (0) Bugcheck on long second.

DEVICE_SPECIFIC=FUNCTION="STOP"

Stop the chip, thereby inducing a transmit timeout on the next transmit request.

DEVICE_SPECIFIC=FUNCTION=("XTMO",VALUE=*setting*)

Set the transmit timeout value in the range 2 to 10000. The actual timeout will be (*setting* − 1) × 250 milliseconds.

DEVICE_SPECIFIC=FUNCTION=("AVER",VALUE=*setting*)

Disable transmit timeout averting if *setting* is 4747.

Set to 4747 to avoid trying to avert a transmit timeout. Setting XTMO to 1 artificially causes many transmit timeouts, which exercises the transmit timeout handling. If not set to 4747, the transmit would have been found to have completed and the timeout would be counted as an "averted" transmit timeout.

DEVICE_SPECIFIC=FUNCTION="OMSG"

Reset number of OPCOM messages.

DEVICE_SPECIFIC=FUNCTION="SYSI"

Send the periodic System ID messages now instead of waiting until expiry of the 8–12-minute System ID timer.

This is a convenient way to get the driver to send two packets.

DEVICE_SPECIFIC=FUNCTION="SYSZ"

Disable the sending of periodic System ID messages.

This is used to disable transmits that might distract from other traffic you are looking at.

DEVICE_SPECIFIC=(FUNCTION="WAIT",VALUE=*timevalue*)

Wait the specified time in nanoseconds, at IPL 8 with the driver port lock held.

This is a convenient way to introduce a delay at IPL 8.

The *timevalue* is 32 bits, limiting the wait to 4 seconds.

DEVICE_SPECIFIC=FUNCTION="CVME"
DEVICE_SPECIFIC=FUNCTION="SVME"

IEEE VLAN Mode Enable (VME) bit set/clear. If clear, the VLAN tag will not be stripped from the packet so it can be seen in the receive buffer.

DEVICE_SPECIFIC=FUNCTION="ARST"

Restart auto-negotiation.

This is used for link handling testing.

DEVICE_SPECIFIC=FUNCTION="FLSC"

Force a link state change interrupt.

DEVICE_SPECIFIC=FUNCTION=("DELA",VALUE=*n*)

Delay the next transmit by *n* 10-millisecond ticks.

DEVICE_SPECIFIC=FUNCTION="XDEL"

Call INI$BRK.

DEVICE_SPECIFIC=FUNCTION="STIM"

Measure statistics time, view result in internal counters "Soft errors".

Chapter 3. VirtIO Ethernet

The VirtIO devices are legacy, transitional, and non-transitional. The VirtIO driver supports transitional and non-transitional devices.

Vendor ID	Sub IDs	Bus	Name	Description
10001AF4	catchall	PCIe/PCI	VirtIO	KVM Virtio (transitional) (VIRTIO-NET-PCI-TRANSITIONAL)
10411AF4	catchall	PCIe/PCI	VirtIO	KVM Virtio (non-transitional) (VIRTIO-NET-PCI-NON-TRANSITIONAL)

The driver is SYS$EVDRIVER.EXE.

There are multiple names for each device, as follows:

- PCI ID repository.
- Vendor names and part numbers.
- Full name assigned by the driver.

  The full name is specified by the "device =" parameter in the entry in SYS$SYSTEM:SYS$CONFIG.DAT and is the same name assigned by the driver. The full name can be found in the list of devices displayed by the **`SEARCH SYS$SYSTEM:SYS$CONFIG.DAT DEVICE,"="/MATCH=AND`** command.

  The full name is also displayed by the LANCP **`SHOW DEVICE/INTERNAL`** command and the SDA **`CLUE CONFIG/ADAPTER`** command.

- Short name assigned by the driver that is used for output requiring a shorter name format.

  The short name is displayed by the LANCP **`SHOW CONFIG`** and **`SHOW DEVICE/INTERNAL`** commands.

3.1. VirtIO Internal Counters

The LANCP command **`SHOW DEVICE/INTERNAL_COUNTERS EIA`** displays the entire set of internal counters maintained by the VirtIO driver. Some counters are special debug counters. These are not displayed unless the additional qualifier **`/DEBUG`** is specified. Counters that are zero are not displayed unless the additional qualifier **`/ZERO`** is specified.

The LAN SDA extension also displays the complete set of internal counters with the **`LAN INTERNAL/DEVICE=EIA`** command.

Note

The definitions of the following counters may change between versions of the driver.

Counters

Device name (full)
Device name (short)

Name of the VirtIO device.

Driver timestamp

Compilation timestamp of the driver.

Driver version

Driver version numbered 1–*n* that is usually identical to the *x*-*n* ID displayed in the **ANALYZE/IMAGE** command output for the driver image. It includes variant information, if applicable. The full driver version includes the target OpenVMS release and is displayed by the SDA **LAN/DEVICE=EIA** command in the quadword driver version field.

Device features

VirtIO features offered by the device.

Driver offered

VirtIO features offered by the driver.

Driver accepted

VirtIO features offered by the device and accepted by the driver.

Device status (last read/written)

Last device status read/written.

Time of last "driver operational"

Time driver reported operational status to the device.

Time of last "device needs reset"

Time device reported that it needs to be reset.

Time of last device status reset

Time device status was last reset.

Device interrupts

Number of times the driver interrupt service routine was called.

Link transitions

Number of link transitions seen by the driver.

Link checks

Number of times the driver checked the current link status.

VCI port usable/unusable events reported

Number of VCI user port events reported.

Device resets

Number of times the device was reset.

Unit inits

Number of unit initializations executed; since unit initialization is only executed once, this counter will be 1.

User start/change/stop requests

Number of user startup and shutdown requests processed by the driver; this is typically one or two requests when a user starts up and one when a user stops.

Transmits queued

Number of transmit requests queued because the link was not available or because too many transmit requests were already outstanding.

Transmit timeouts

Number of times the driver has timed out a transmit, reset the device, and completed outstanding I/O with error status.

Transmit timeouts (averted)

Number of transmit timeouts averted by a final check for transmit completion.

Transmit chained (buffer address)

Number of chained transmits where the chaining is via SVA buffer address.

Transmit chained (SVAPTE_SVA)

Number of chained transmits where the chaining is via a SVAPTE_SVA-style buffer.

Transmit chained (IOBD/Extent)

Number of chained transmits where the chaining is via an IOBD/Extent-style buffer.

Transmit errors (too few segments)

Number of transmit requests completed with error status (SS$_INCSEGTRA) because the application did not specify the transmit buffer completely.

Transmit informationals (zero-byte segments)

Number of transmit chain segments which specified a zero-byte length segment.

Transmit copies (too many segments)

Number of transmit requests that exceeded the maximum number of chain segments the driver can handle; the driver then copied some of them to a temporary buffer so that it could transmit the packet.

Transmit copy failures

Number of transmit failures caused by too few chain segments when coalescing a request to reduce the number of segments needed.

Jumbo transmits issued

Number of transmit requests with a packet length exceeding 1514 bytes (excluding CRC).

Jumbo receives done

Number of jumbo packets received.

Chained receives completed

Number of receives (jumbo packets) completed by the driver that span multiple receive buffers.

Soft errors

Number of times errors were recovered in the driver by resetting and reinitializing the device without notifying user applications.

Rescheduled forks (too long in fork)

Number of times that a rescheduled fork was done. In transmit and receive processing, the driver limits the amount of time spent in the fork process before rescheduling.

Standard packet size (bytes)

Ethernet packet size (1518 bytes).

Jumbo packet size (bytes)

Jumbo packet size (9018 bytes).

RX (Receive) commands issued

Receive filtering commands issued to the device.

PHA (Physical Address) commands issued

Set MAC address commands issued to the device.

MCA (Multicast Address) commands issued

Multicast filtering commands issued to the device.

Requested link state

Speed and duplex mode requested by a user.

Current link state

Current link state.

Driver flags

Driver flags.

Driver state

Current driver state, as follows:

0 – Driver state is undefined.
1 – Driver is running, but the link is down. Completing transmits with error status.
2 – Driver is running, but the link is down. Queuing transmits for now.
4 – Driver is running, and the link is up. Processing transmits and receives.
8 – Device is not usable.

LAN_FLAGS system parameter

Current value of the LAN_FLAGS system parameter.

LSB size

Total size of the LSB structure.

Interrupt mode

Unknown, IOSAPIC, MSI, MSI-X, or IOAPIC.

Receive ring PA
Transmit ring PA
Control ring PA
Receive ring VA
Transmit ring VA
Control ring VA
Receive ring size
Transmit ring size
Control ring size
Receive ring notify offset
Transmit ring notify offset
Control ring notify offset
Notify offset multiplier

Physical address (PA) and virtual address (VA), size, and notification offset of the driver ring structures.

PCI capability offsets

PCI capability values

Common config VA
Notify config VA
ISR config VA
Device config VA

Configuration structure VAs.

RX mode buffer VA (active)
MAC address buffer VA (active)
MCA address buffer VA (active)

Configuration buffer VAs.

Transmit time limit

Transmit time limit in seconds after which a timeout is declared.

Timer routine interval

Resolution of the transmit timer (the real timeout is limit + interval).

Time Stamps

Current uptime

Current system uptime.

Last reset

Last time the device was reset.

Last link state change interrupt

Last time a link state change interrupt was fielded.

Previous link state change interrupt

Previous time a link state change interrupt was fielded.

Last link up

Last time a link up transition occurred.

Last link down

Last time a link down transition occurred.

Total link uptime

Total time the link has been up.

Total link downtime

Total time the link has been down.

Time of last uplink period
Time of last uplink period-1
Time of last uplink period-2
Time of last uplink period-3
Time of last uplink period-4
Time of last uplink period-5
Time of last uplink period-6
Time of last uplink period-7
Time of last uplink period-8
Time of last uplink period-9
Time of last uplink period-10
Time of last uplink period-11
Time of last uplink period-12
Time of last uplink period-13
Time of last uplink period-14

History of the last 15 link uptimes, displaying the length of each uptime period.

Last transmit timeout

Time of the last transmit timeout.

Last transmit timeout (averted)

Time last averted a transmit timeout.

Last soft error

Time of the last soft error.

Last unrecognized unicast packet

Time of the last unrecognized unicast packet received.

Last unrecognized unicast packet header

First 24 bytes of the last unicast packet received that was discarded because receive address filtering did not result in a matching user.

Last unrecognized multicast packet

Time of the last unrecognized multicast packet received.

Last unrecognized multicast packet header

First 24 bytes of the last multicast packet received that was discarded because receive address filtering did not result in a matching user.

PCI information

- PCI config registers
- Common CFG data
- Device CFG data

Receive Buffers

Minimum buffers requested

Minimum number of receive buffers as set by default or by management request (SETMODE or LANCP command). This is used to determine the "Current minimum limit".

Maximum buffers requested

Maximum number of receive buffers as set by default or by management request (SETMODE or LANCP command). This is used to determine the "Current maximum limit".

Current minimum limit

Minimum number of receive buffers as determined by the driver and by management request, "Minimum buffers requested". The driver will not allow the number of receive buffers in its receive queues to drop below this value. This does not include receive buffers transferred to applications (and not yet returned).

Current maximum limit

Maximum number of receive buffers as determined by the driver and by management request, "Maximum buffers requested". The driver will deallocate receive buffers until the number allocated does not exceed this maximum. When applications return receive buffers, the number may exceed this maximum, at which point they are deallocated. Note that the driver allows this maximum to be exceeded slightly to limit allocation and deallocation activity. See "Target number of buffers maximum" below.

Current number of buffers

Current number of receive buffers owned by the driver.

Target number of buffers

Desired number of receive buffers. As receive activity does not result in lost packets, this number is gradually decreased toward the "Current minimum limit". If packets are lost, this number is dramatically increased toward the "Current maximum limit". When the driver allocates receive buffers, it allocates them until reaching this target number.

Target number of buffers maximum

Maximum desired number of receive buffers. When an application returns a buffer to the driver, if the current number does not exceed this target maximum, the buffer is kept by the driver. Otherwise, it is deallocated.

Fork Delay (after scheduled)

To help determine whether the buffering requirements of the driver and the device are sufficient for the system configuration, the driver records the amount of time from fork scheduled to the time the fork is actually run. The data is recorded in 10-millisecond increments from 10 to 310 milliseconds.

This data can be used in conjunction with the number of packets discarded due to insufficient buffers to determine whether the buffering settings of the driver (minimum and maximum receive buffers) and the amount of buffering on the device are sufficient for normal operation. If packets are being discarded, the buffering should be increased until the number of packets lost is minimal.

Transmit Time

To help understand the operation of the system, driver, and device, the driver records the time that each transmit buffer is given to the device. When the transmit completes, the driver calculates the elapsed time, creating a histogram of transmit times from 10 to 310 milliseconds.

Receive completion time

To help understand the operation of the system and driver, the driver records the time taken to process each receive buffer. When the receive completes, the driver records the time that processing started. Then it delivers the packet to the user application. When the user returns it, the driver calculates this completion time, creating a histogram of receive times from 10 to 150 milliseconds.

One second timer time

The driver maintains a one-second timer (that runs four times a second). It expects the timer routine to be called close to the timer interval. By looking at the time difference between when a timer is expected to be called and when it actually is called, it is possible to understand the operation of the

system and driver. The variance time is used to create a histogram of receive times from 10 to 150 milliseconds.

Driver Messages

Console messages issued by the driver.

3.2. VirtIO Device-Specific Functions

The **SET DEVICE /DEVICE_SPECIFIC=(FUNCTION="FUNC",VALUE=(V1,V2,...))** LANCP command provides a mechanism to issue device-specific functions to the VirtIO driver. While some functions are common to many LAN drivers, most are primarily useful in a diagnostic context and are not typically included in LANCP command documentation.

For further information on the LANCP utility and LANCP commands, refer to the relevant section in the *VSI OpenVMS System Management Utilities Reference Manual, Volume I: A–L* [https://docs.vmssoftware.com/vsi-openvms-system-management-utilities-reference-manual-volume-i-a-l/#d0e28590] or the LANCP online Help.

Notes

The function name is always four characters in length.

The command requires the SYSPRV privilege.

The definitions of the following functions may change between versions of the driver.

Device-Specific Functions

DEVICE_SPECIFIC=FUNCTION=("RCOR",VALUE=%X*ptyvalue*)
DEVICE_SPECIFIC=FUNCTION=("XCOR",VALUE=%X*ptyvalue*)

Corrupt a transmit or receive packet. When this function is executed, the protocol type is specified; if it matches on a transmit or receive packet, the next packet seen is corrupted by changing the last bit in the VCRP or CXB, XORing it with a 1. The protocol type is then cleared so that the corruption is not done again.

The following example shows how to corrupt a PEDRIVER transmit packet:

```
$ MCR LANCP SET DEVICE/TRACE=(MASK=XMTISSUE) EIA
$ MCR LANCP SET DEVICE/DEVICE_SPECIFIC=(FUNCTION="XCOR", VALUE=%x760) EIA
```

The following example shows how to corrupt a PEDRIVER receive packet:

```
$ MCR LANCP SET DEVICE/TRACE=(MASK=RCVDONE) EIA
$ MCR LANCP SET DEVICE/DEVICE_SPECIFIC=(FUNCTION="RCOR", VALUE=%x760) EIA
```

DEVICE_SPECIFIC=FUNCTION=("LOSE",VALUE=%x*aabbccdd*)

Lose some packets by corrupting the header of the packet. Corrupt packet 0x*aabbccdd*, where *aa* is seconds to corrupt, *bb* is number to corrupt, *cc* is byte number 0–255, and *dd* is what to corrupt with.

For example, to send 32 transmit packets elsewhere (no more than 16 seconds), change the last byte of the destination address to 0x44. For example:

```
$ MCR LANCP SET DEVICE/TRACE=(MASK=XMTISSUE) EIA
$ MCR LANCP SET DEVICE/DEVICE_SPECIFIC=(FUNCTION="LOSE", VALUE=%x10200544) EIA
```

To lose just one packet, change the first byte of the packet to 0x00. For example:

```
$ MCR LANCP SET DEVICE/TRACE=(MASK=XMTISSUE) EIA
$ MCR LANCP SET DEVICE/DEVICE_SPECIFIC=FUNCTION="LOSE" EIA
```

To lose 255 packets for the rest of the 10-millisecond tick, change the first byte of the packet to 0x00. For example:

```
$ MCR LANCP SET DEVICE/TRACE=(MASK=XMTISSUE) EIA
$ MCR LANCP SET DEVICE/DEVICE_SPECIFIC=(FUNCTION="LOSE",VALUE=%xFF0000) EIA
```

DEVICE_SPECIFIC=FUNCTION=("FCOP",VALUE=*setting*)

Force transmit chaining coalescing to exercise this rarely-used code path.

Enable (1) or disable (0) this function.

DEVICE_SPECIFIC=FUNCTION=("PROM",VALUE=*setting*)

Enable (1) or disable (0) force promiscuous mode, then reset the device.

DEVICE_SPECIFIC=FUNCTION=("XCOA",VALUE=*setting*)

Allow (1) or disallow (0) transmit interrupt coalescing.

DEVICE_SPECIFIC=FUNCTION="SOFT"

Force a non-fatal (soft) error. The device is restarted.

DEVICE_SPECIFIC=(FUNCTION="PRES",VALUE=*periodicmask*)

Schedule periodic random resets. The value supplied is AND'ed with (RSCC × 69 069) and, if the result is 3, a reset is done. For example, if the value supplied is 7, a reset is done every 8 seconds on average.

A value of zero disables this function, and a nonzero value enables it.

This is used to test error handling.

DEVICE_SPECIFIC=FUNCTION=("ALLM",VALUE=*setting*)

Enable (1) or disable (0) all messages flag.

DEVICE_SPECIFIC=FUNCTION=("FLIN",VALUE=*setting*)

Force link indication. If the value is 0 (disabled), do not force the link state.

If the value is 1, force the link to be up.

If the value is 2, force the link to be down.

DEVICE_SPECIFIC=FUNCTION=("XTOB",VALUE=*setting*)

Enable (1) or disable (0) Bugcheck on transmit timeout.

DEVICE_SPECIFIC=FUNCTION=("XLON",VALUE=*setting*)

Enable (1) or disable (0) Bugcheck on long transmit completion.

DEVICE_SPECIFIC=FUNCTION=("FLON",VALUE=*setting*)

Enable (1) or disable (0) Bugcheck on long fork.

DEVICE_SPECIFIC=FUNCTION=("SLON",VALUE=*setting*)

Enable (1) or disable (0) Bugcheck on long second.

DEVICE_SPECIFIC=FUNCTION=("XTMO",VALUE=*setting*)

Set the transmit timeout value in the range 2 to 10000. The actual timeout will be (*setting* − 1) × 250 milliseconds.

DEVICE_SPECIFIC=FUNCTION=("AVER",VALUE=*setting*)

Disable transmit timeout averting if *setting* is 4747.

Set to 4747 to avoid trying to avert a transmit timeout. Setting XTMO to 1 artificially causes many transmit timeouts, which exercises the transmit timeout handling. If not set to 4747, the transmit would have been found to have completed and the timeout would be counted as an "averted" transmit timeout.

DEVICE_SPECIFIC=FUNCTION="OMSG"

Reset number of OPCOM messages.

DEVICE_SPECIFIC=FUNCTION="SYSI"

Send the periodic System ID messages now instead of waiting until expiry of the 8–12-minute System ID timer.

This is a convenient way to get the driver to send two packets.

DEVICE_SPECIFIC=FUNCTION="SYSZ"

Disable the sending of periodic System ID messages.

This is used to disable transmits that might distract from other traffic you are looking at.

DEVICE_SPECIFIC=(FUNCTION="WAIT",VALUE=*timevalue*)

Wait the specified time in nanoseconds, at IPL 8 with the driver port lock held.

This is a convenient way to introduce a delay at IPL 8.

The *timevalue* is 32 bits, limiting the wait to 4 seconds.

DEVICE_SPECIFIC=FUNCTION=("DELA",VALUE=*n*)

Delay the next transmit by *n* 10-millisecond ticks.

DEVICE_SPECIFIC=FUNCTION="XDEL"

Call INI$BRK.

DEVICE_SPECIFIC=FUNCTION="QDIS"

Display queue data on the console.

Chapter 4. VMXNET3 Ethernet

VMXNET3 is supported on VMware by Broadcom-developed hypervisors.

Vendor ID	Sub IDs	Bus	Name	Description
07B015AD	catchall	PCIe/PCI	VMXNET3	VMware VMXNET3

The driver is SYS$VMXNET3.EXE.

There are multiple names for the device, as follows:

- PCI ID repository.
- Vendor names and part numbers.
- Full name assigned by the driver.

  The full name is specified by the "device =" parameter in the entry in SYS$SYSTEM:SYS$CONFIG.DAT and is the same name assigned by the driver. The full name can be found in the list of devices displayed by the **`SEARCH SYS$SYSTEM:SYS$CONFIG.DAT DEVICE,"="/MATCH=AND`** command.

  The full name is also displayed by the LANCP **`SHOW DEVICE/INTERNAL`** command and the SDA **`CLUE CONFIG/ADAPTER`** command.

- Short name assigned by the driver that is used for output requiring a shorter name format.

  The short name is displayed by the LANCP **`SHOW CONFIG`** and **`SHOW DEVICE/INTERNAL`** commands.

4.1. VMXNET3 Internal Counters

The LANCP command **`SHOW DEVICE/INTERNAL_COUNTERS EIA`** displays the entire set of internal counters maintained by the VMXNET3 driver. Some counters are special debug counters. These are not displayed unless the additional qualifier **`/DEBUG`** is specified. Counters that are zero are not displayed unless the additional qualifier **`/ZERO`** is specified.

The LAN SDA extension also displays the complete set of internal counters with the **`LAN INTERNAL/DEVICE=EIA`** command.

Note

The definitions of the following counters may change between versions of the driver.

Counters

Device name (full)
Device name (short)

Name of the VMXNET3 device.

Driver timestamp

Compilation timestamp of the driver.

Driver version

Driver version numbered 1–*n* that is usually identical to the *x*-*n* ID displayed in the **ANALYZE/IMAGE** command output for the driver image. It includes variant information, if applicable. The full driver version includes the target OpenVMS release and is displayed by the SDA **LAN/DEVICE=EIA** command in the quadword driver version field.

Device revision

As reported by the device.

Device interrupts

Number of times the driver interrupt service routine was called.

Link transitions

Number of link transitions seen by the driver.

Link checks

Number of times the driver checked the current link status.

VCI port usable/unusable events reported

Number of VCI user port events reported.

Device resets

Number of times the device was reset.

CSR Base Address, BAR0
CSR Base Address, BAR1
CSR Base Address, BAR2

Values of the PCI Configuration Space Base Address Registers (BAR).

Unit inits

Number of unit initializations executed; since unit initialization is only executed once, this counter will be 1.

User start/change/stop requests

Number of user startup and shutdown requests processed by the driver; this is typically one or two requests when a user starts up and one when a user stops.

Transmits queued

Number of transmit requests queued because the link was not available or because too many transmit requests were already outstanding.

Transmit timeouts

Number of times the driver has timed out a transmit, reset the device, and completed outstanding I/O with error status.

Transmit timeouts (averted)

Number of transmit timeouts averted by a final check for transmit completion.

Transmit chained (buffer address)

Number of chained transmits where the chaining is via SVA buffer address.

Transmit chained (SVAPTE_SVA)

Number of chained transmits where the chaining is via a SVAPTE_SVA-style buffer.

Transmit chained (IOBD/Extent)

Number of chained transmits where the chaining is via an IOBD/Extent-style buffer.

Transmit errors (too few segments)

Number of transmit requests completed with error status (SS$_INCSEGTRA) because the application did not specify the transmit buffer completely.

Transmit informationals (zero-byte segments)

Number of transmit chain segments which specified a zero-byte length segment.

Transmit copies (too many segments)

Number of transmit requests that exceeded the maximum number of chain segments the driver can handle; the driver then copied some of them to a temporary buffer so that it could transmit the packet.

Transmit copy failures

Number of transmit failures caused by too few chain segments when coalescing a request to reduce the number of segments needed.

Jumbo transmits issued

Number of transmit requests with a packet length exceeding 1514 bytes (excluding CRC).

Jumbo receives done

Number of jumbo packets received.

Chained receives completed

Number of receives (jumbo packets) completed by the driver that span multiple receive buffers.

Receive errors (unwanted segments)

Number of received segments marked as unwanted.

Receive indices out of sync

Number of times receive index was out of sync.

Soft errors

Number of times errors were recovered in the driver by resetting and reinitializing the device without notifying user applications.

Rescheduled forks (too long in fork)

Number of times that a rescheduled fork was done. In transmit and receive processing, the driver limits the amount of time spent in the fork process before rescheduling.

Standard packet size (bytes)

Ethernet packet size (1518 bytes).

Jumbo packet size (bytes)

Jumbo packet size (9018 bytes).

Requested link state

Speed and duplex mode requested by a user.

Current link state

Current link state.

Driver flags

Driver flags.

Driver state

Current driver state, as follows:

0 – Driver state is undefined.
1 – Driver is running, but the link is down. Completing transmits with error status.
2 – Driver is running, but the link is down. Queuing transmits for now.
4 – Driver is running, and the link is up. Processing transmits and receives.
8 – Device is not usable.

LAN_FLAGS system parameter

Current value of the LAN_FLAGS system parameter.

LSB size

Total size of the LSB structure.

Shared Data PA
Shared Data VA
Shared Data length
Receive ring VA
Receive completion ring VA
Transmit ring VA
Transmit completion ring VA
Receive ring generation
Receive completion ring generation
Transmit ring generation
Transmit completion ring generation

Shared structure context.

Interrupt mode

Unknown, IOSAPIC, MSI, MSI-X, or IOAPIC.

Transmit time limit

Transmit time limit in seconds after which a timeout is declared.

Timer routine interval

Resolution of the transmit timer (the real timeout is limit + interval).

Time Stamps

Current uptime

Current system uptime.

Last reset

Last time the device was reset.

Last link state change interrupt

Last time a link state change interrupt was fielded.

Previous link state change interrupt

Previous time a link state change interrupt was fielded.

Last link up

Last time a link up transition occurred.

Last link down

Last time a link down transition occurred.

Total link uptime

Total time the link has been up.

Total link downtime

Total time the link has been down.

Time of last uplink period
Time of last uplink period-1
Time of last uplink period-2
Time of last uplink period-3
Time of last uplink period-4
Time of last uplink period-5
Time of last uplink period-6
Time of last uplink period-7
Time of last uplink period-8
Time of last uplink period-9
Time of last uplink period-10
Time of last uplink period-11
Time of last uplink period-12
Time of last uplink period-13
Time of last uplink period-14

History of the last 15 link uptimes, displaying the length of each uptime period.

Last transmit timeout

Time of the last transmit timeout.

Last transmit timeout (averted)

Time last averted a transmit timeout.

Last soft error

Time of the last soft error.

Last CRC error

Time of the last CRC error.

Last receive error

First 24 bytes of the last packet with a receive error.

Last unrecognized unicast packet

Time of the last unrecognized unicast packet received.

Last unrecognized unicast packet header

First 24 bytes of the last unicast packet received that was discarded because receive address filtering did not result in a matching user.

Last unrecognized multicast packet

Time of the last unrecognized multicast packet received.

Last unrecognized multicast packet header

First 24 bytes of the last multicast packet received that was discarded because receive address filtering did not result in a matching user.

Device Registers (read/wrote)

List of device registers and the value last read or written.

PCI Config Registers

List of PCI Configuration Space registers and the values recorded.

Receive Buffers

Minimum buffers requested

Minimum number of receive buffers as set by default or by management request (SETMODE or LANCP command). This is used to determine the "Current minimum limit".

Maximum buffers requested

Maximum number of receive buffers as set by default or by management request (SETMODE or LANCP command). This is used to determine the "Current maximum limit".

Current minimum limit

Minimum number of receive buffers as determined by the driver and by management request, "Minimum buffers requested". The driver will not allow the number of receive buffers in its receive queues to drop below this value. This does not include receive buffers transferred to applications (and not yet returned).

Current maximum limit

Maximum number of receive buffers as determined by the driver and by management request, "Maximum buffers requested". The driver will deallocate receive buffers until the number allocated does not exceed this maximum. When applications return receive buffers, the number may exceed this maximum, at which point they are deallocated. Note that the driver allows this maximum to be exceeded slightly to limit allocation and deallocation activity. See "Target number of buffers maximum" below.

Current number of buffers

Current number of receive buffers owned by the driver.

Target number of buffers

Desired number of receive buffers. As receive activity does not result in lost packets, this number is gradually decreased toward the "Current minimum limit". If packets are lost, this number is dramatically increased toward the "Current maximum limit". When the driver allocates receive buffers, it allocates them until reaching this target number.

Target number of buffers maximum

Maximum desired number of receive buffers. When an application returns a buffer to the driver, if the current number does not exceed this target maximum, the buffer is kept by the driver. Otherwise, it is deallocated.

Fork Delay (after scheduled)

To help determine whether the buffering requirements of the driver and the device are sufficient for the system configuration, the driver records the amount of time from fork scheduled to the time the fork is actually run. The data is recorded in 10-millisecond increments from 10 to 310 milliseconds.

This data can be used in conjunction with the number of packets discarded due to insufficient buffers to determine whether the buffering settings of the driver (minimum and maximum receive buffers) and the amount of buffering on the device are sufficient for normal operation. If packets are being discarded, the buffering should be increased until the number of packets lost is minimal.

Transmit Time

To help understand the operation of the system, driver, and device, the driver records the time that each transmit buffer is given to the device. When the transmit completes, the driver calculates the elapsed time, creating a histogram of transmit times from 10 to 310 milliseconds.

Receive completion time

To help understand the operation of the system and driver, the driver records the time taken to process each receive buffer. When the receive completes, the driver records the time that processing

started. Then it delivers the packet to the user application. When the user returns it, the driver calculates this completion time, creating a histogram of receive times from 10 to 150 milliseconds.

One second timer time

The driver maintains a one-second timer (that runs four times a second). It expects the timer routine to be called close to the timer interval. By looking at the time difference between when a timer is expected to be called and when it actually is called, it is possible to understand the operation of the system and driver. The variance time is used to create a histogram of receive times from 10 to 150 milliseconds.

Driver Messages

Console messages issued by the driver.

4.2. VMXNET3 Device-Specific Functions

The **`SET DEVICE /DEVICE_SPECIFIC=(FUNCTION="FUNC",VALUE=(V1,V2,...))`** LANCP command provides a mechanism to issue device-specific functions to the VMXNET3 driver. While some functions are common to many LAN drivers, most are primarily useful in a diagnostic context and are not typically included in LANCP command documentation.

For further information on the LANCP utility and LANCP commands, refer to the relevant section in the *VSI OpenVMS System Management Utilities Reference Manual, Volume I: A–L* [https://docs.vmssoftware.com/vsi-openvms-system-management-utilities-reference-manual-volume-i-a-l/#d0e28590] or the LANCP online Help.

Notes

The function name is always four characters in length.

The command requires the SYSPRV privilege.

The definitions of the following functions may change between versions of the driver.

Device-Specific Functions

DEVICE_SPECIFIC=FUNCTION=("RCOR",VALUE=%X*ptyvalue*)
DEVICE_SPECIFIC=FUNCTION=("XCOR",VALUE=%X*ptyvalue*)

Corrupt a transmit or receive packet. When this function is executed, the protocol type is specified; if it matches on a transmit or receive packet, the next packet seen is corrupted by changing the last bit in the VCRP or CXB, XORing it with a 1. The protocol type is then cleared so that the corruption is not done again.

The following example shows how to corrupt a PEDRIVER transmit packet:

```
$ MCR LANCP SET DEVICE/TRACE=(MASK=XMTISSUE) EIA
$ MCR LANCP SET DEVICE/DEVICE_SPECIFIC=(FUNCTION="XCOR", VALUE=%x760) EIA
```

The following example shows how to corrupt a PEDRIVER receive packet:

```
$ MCR LANCP SET DEVICE/TRACE=(MASK=RCVDONE) EIA
$ MCR LANCP SET DEVICE/DEVICE_SPECIFIC=(FUNCTION="RCOR", VALUE=%x760) EIA
```

DEVICE_SPECIFIC=FUNCTION=("LOSE",VALUE=%x*aabbccdd*)

Lose some packets by corrupting the header of the packet. Corrupt packet 0x*aabbccdd*, where *aa* is seconds to corrupt, *bb* is number to corrupt, *cc* is byte number 0–255, and *dd* is what to corrupt with.

For example, to send 32 transmit packets elsewhere (no more than 16 seconds), change the last byte of the destination address to 0x44. For example:

```
$ MCR LANCP SET DEVICE/TRACE=(MASK=XMTISSUE) EIA
$ MCR LANCP SET DEVICE/DEVICE_SPECIFIC=(FUNCTION="LOSE", VALUE=%x10200544) EIA
```

To lose just one packet, change the first byte of the packet to 0x00. For example:

```
$ MCR LANCP SET DEVICE/TRACE=(MASK=XMTISSUE) EIA
$ MCR LANCP SET DEVICE/DEVICE_SPECIFIC=FUNCTION="LOSE" EIA
```

To lose 255 packets for the rest of the 10-millisecond tick, change the first byte of the packet to 0x00. For example:

```
$ MCR LANCP SET DEVICE/TRACE=(MASK=XMTISSUE) EIA
$ MCR LANCP SET DEVICE/DEVICE_SPECIFIC=(FUNCTION="LOSE",VALUE=%xFF0000) EIA
```

DEVICE_SPECIFIC=FUNCTION=("FCOP",VALUE=*setting*)

Force transmit chaining coalescing to exercise this rarely-used code path.

Enable (1) or disable (0) this function.

DEVICE_SPECIFIC=FUNCTION=("PROM",VALUE=*setting*)

Enable (1) or disable (0) force promiscuous mode, then reset the device.

DEVICE_SPECIFIC=FUNCTION="SOFT"

Force a non-fatal (soft) error. The device is restarted.

DEVICE_SPECIFIC=(FUNCTION="PRES",VALUE=*periodicmask*)

Schedule periodic random resets. The value supplied is AND'ed with (RSCC × 69 069) and, if the result is 3, a reset is done. For example, if the value supplied is 7, a reset is done every 8 seconds on average.

A value of zero disables this function, and a nonzero value enables it.

This is used to test error handling.

DEVICE_SPECIFIC=FUNCTION=("ALLM",VALUE=*setting*)

Enable (1) or disable (0) all messages flag.

DEVICE_SPECIFIC=FUNCTION=("FLIN",VALUE=*setting*)

Force link indication. If the value is 0 (disabled), do not force the link state.

If the value is 1, force the link to be up.

If the value is 2, force the link to be down.

DEVICE_SPECIFIC=FUNCTION=("XTOB",VALUE=*setting*)

Enable (1) or disable (0) Bugcheck on transmit timeout.

DEVICE_SPECIFIC=FUNCTION=("XLON",VALUE=*setting*)

Enable (1) or disable (0) Bugcheck on long transmit completion.

DEVICE_SPECIFIC=FUNCTION=("FLON",VALUE=*setting*)

Enable (1) or disable (0) Bugcheck on long fork.

DEVICE_SPECIFIC=FUNCTION=("SLON",VALUE=*setting*)

Enable (1) or disable (0) Bugcheck on long second.

DEVICE_SPECIFIC=FUNCTION=("XTMO",VALUE=*setting*)

Set the transmit timeout value in the range 2 to 10000. The actual timeout will be (*setting* − 1) × 250 milliseconds.

DEVICE_SPECIFIC=FUNCTION=("AVER",VALUE=*setting*)

Disable transmit timeout averting if *setting* is 4747.

Set to 4747 to avoid trying to avert a transmit timeout. Setting XTMO to 1 artificially causes many transmit timeouts, which exercises the transmit timeout handling. If not set to 4747, the transmit would have been found to have completed and the timeout would be counted as an "averted" transmit timeout.

DEVICE_SPECIFIC=FUNCTION="OMSG"

Reset number of OPCOM messages.

DEVICE_SPECIFIC=FUNCTION="SYSI"

Send the periodic System ID messages now instead of waiting until expiry of the 8–12-minute System ID timer.

This is a convenient way to get the driver to send two packets.

DEVICE_SPECIFIC=FUNCTION="SYSZ"

Disable the sending of periodic System ID messages.

This is used to disable transmits that might distract from other traffic you are looking at.

DEVICE_SPECIFIC=(FUNCTION="WAIT",VALUE=*timevalue*)

Wait the specified time in nanoseconds, at IPL 8 with the driver port lock held.

This is a convenient way to introduce a delay at IPL 8.

The *timevalue* is 32 bits, limiting the wait to 4 seconds.

DEVICE_SPECIFIC=FUNCTION=("DELA",VALUE=*n*)

Delay the next transmit by *n* 10-millisecond ticks.

DEVICE_SPECIFIC=FUNCTION="XDEL"

Call INI$BRK.

DEVICE_SPECIFIC=FUNCTION="REGS"

Display device registers on the console.

DEVICE_SPECIFIC=FUNCTION="QSTA"

Update queue status from the device.

DEVICE_SPECIFIC=FUNCTION="BARS"

Display the BAR registers on the console.