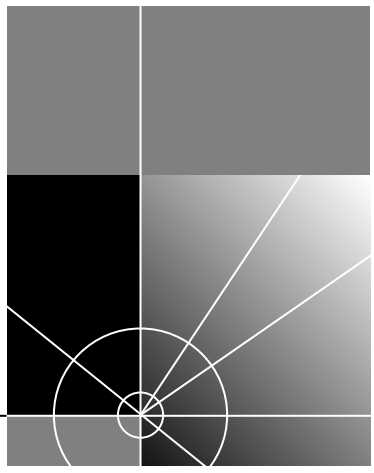
# 3C90x and 3C90xB NICs Technical Reference

**3Com® EtherLink® XL and Fast EtherLink XL PCI network interface cards**



**http://www.3com.com/**

Part Number: 89-0766-000
Published August 1998

**3Com Corporation**
**5400 Bayfront Plaza**
**Santa Clara, California**
**95052-8145**

# CONTENTS

## 3 OPERATION



## 4 CONFIGURATION

## 5 EEPROM



## 6 DOWNLOAD AND TRANSMISSION

## 10 COMMAND REGISTER



## 11 AUTO-NEGOTIATION AND MII REGISTERS

## 12 OTHER REGISTERS

## A AUTOSELECT PSEUDO CODE



## B PROGRAMMING THE MII MANAGEMENT INTERFACE



## C FRAME FORMATS



## D ERRATA LIST AND SOFTWARE SOLUTIONS



## INDEX

## INDEX OF REGISTERS

## INDEX OF BITS

# FIGURES

# TABLES

# 1 INTRODUCTION

This technical reference describes the basic architecture and defines the programming interface of 3Com® EtherLink® XL and Fast EtherLink XL network interface cards (NICs). The NIC models are listed in Table 1 and Table 2.

**Table 1** 3C90x NICs

| Model | Speed | Media Type | Cable | Connector |
|---|---|---|---|---|
| 3C900-TPO | 10 Mbps | 10BASE-T | Two-pair Category 3, 4, or 5 UTP | RJ-45 |
| 3C900-COMBO | 10 Mbps | 10BASE-T | Two-pair Category 3, 4, or 5 UTP | RJ-45 |
| | | 10BASE-5 | Thick Ethernet coaxial | AUI |
| | | 10BASE-2 | Thin Ethernet coaxial | BNC |
| 3C905-TX | 10 /100 Mbps | 10BASE-T/ 100BASE-TX | Two-pair Category 3, 4, or 5 UTP/ Two-pair Category 5 UTP | RJ-45 |
| 3C905-T4 | 10/100 Mbps | 10BASE-T/ 100BASE-T4 | Two-pair Category 3, 4, or 5 UTP/ Four-pair Category 5 UTP | RJ-45 |

**Table 2** 3C90xB NICs

| Model | Speed | Media Type | Cable | Connector |
|---|---|---|---|---|
| 3C900B-TPO | 10 Mbps | 10BASE-T | Two-pair Category 3, 4, or 5 UTP | RJ-45 |
| 3C900B-TPC | 10 Mbps | 10BASE-T | Two-pair Category 3, 4, or 5 UTP | RJ-45 |
| | | 10BASE-2 | Thin Ethernet coaxial | BNC |
| 3C900B-COMBO | 10 Mbps | 10BASE-T | Two-pair Category 3, 4, or 5 UTP | RJ-45 |
| | | 10BASE-5 | Thick Ethernet coaxial | AUI |
| | | 10BASE-2 | Thin Ethernet coaxial | BNC |
| 3C905B-TX | 10/100 Mbps | 10BASE-T/ 100BASE-TX | Two-pair Category 5 UTP, or STP | RJ-45 |
| 3C905B-TX-NM | 10/100 Mbps | 10BASE-T/ 100BASE-TX | Two-pair Category 5 UTP, or STP | RJ-45 |
| 3C900B-FL | 10 Mbps | 10BASE-FL | Short-wavelength fiber-optic (850 nm): 50 µ/125 µ and 62.5 µ/125 µ multimode fiber | ST |
| 3C905B-FX | 100 Mbps | 100BASE-FX | Long-wavelength fiber-optic (1300 nm): 50 µ/125 µ and 62.5 µ/125 µ multimode fiber | SC |

## 3C90x NIC Features

3C90x NICs have these features:

- Multipacket, multifragment scatter operations for uploads
- Multipacket, multifragment gather operations for downloads
- Simultaneous upload and download operations
- On-chip RAM that can be used instead of external RAM

## 3C90xB NIC Features

3C90xB NICs have these additional features:

- 2 KB transmit FIFO and 2 KB receive FIFO.
- True dual-channel DMA engine.
- Enhanced scatter-gather engines that reduce the number of I/O operations required to support data transfers.
- A download-scheduling mechanism that allows a packet to be downloaded at some specific future time. For example, download scheduling can be used to support video or audio streams over a LAN, or to avoid overflowing a switch's buffers when the switch is communicating with a lower-speed device.
- A hash filter that provides better multicast packet handling.
- Support for VLANs and IEEE 802.3x flow control functions.
- Support for IEEE 802.3u auto-negotiation (10BASE-T and 100BASE-TX).
- Support for PC '97 guidelines, including ACPI Power Management.
- Support for wake-up events (except 3C900B and 3C905B-TX-NM NICs).
- Improved bus master efficiency through use of optimal PCI memory commands and support of larger burst lengths.
- TCP/IP checksum features.
- Reduced I/O operations.
- Direct register access to BIOS ROM.
- An integrated 100 Mbps PHY that eliminates the need for an external 100 Mbps transceiver (40-0476-001 and 40-0483-00x ASICs only).

*There are three different versions of the ASIC on the 3C905B-TX NIC. These three ASICs function identically except for the PHY portion, which is proprietary to each ASIC. As a result, the MII register layouts differ on the three ASICs. See Chapter 11 for information on the MII register layouts of each ASIC.*

For more information on the ASICs, see "ASICs" in Chapter 2.

## About This Technical Reference

This technical reference contains information that software engineers, independent software developers, and test engineers can use when writing device drivers, diagnostic programs, and production test software for 3C90x and 3C90xB NICs.

Specifications in this technical reference apply to all 3C90x and 3C90xB NICs unless the text designates a specific model or type (for example, 3C900 NIC or 3C90xB NIC).

### Decimal, Binary, and Hexadecimal Numbers

Unless otherwise indicated in the text, all values are decimal. The following are exceptions:

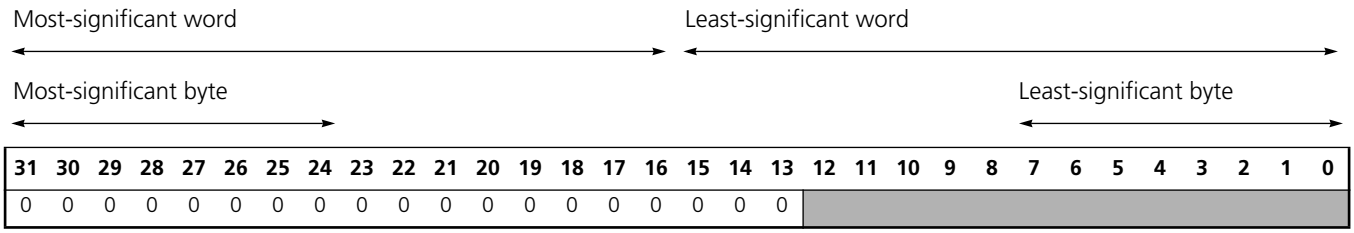
- Binary values are indicated with the character "b" appended to the value (for example, 234b).
- Hexadecimal values are indicated with the character "h" appended to the value (for example, 234h).

### Terms and Acronyms

The following terms and acronyms are used in this reference:

| Term or Acronym | Meaning |
|---|---|
| BIST | Built-in self test. |
| Byte | An 8-bit wide quantity of data. |
| Double word (dword) | A 32-bit wide quantity of data (4 bytes). |
| Download | The process of transferring transmit data from system memory to the NIC. |
| DPD | Download packet descriptor. |
| FLP | Fast link pulse. |
| FSH | Frame start header. |
| Indication | The reporting of any interesting event on the NIC. Any indication may be configured to cause an interrupt. |
| Interrupt | The actual assertion of the host machine's interrupt signal. |
| MII | Media-Independent Interface. |
| NIC | Network interface card. |
| NOS | Network operating system. |
| PEROM | Programmable and erasable read-only memory. |
| PHY | IEEE designation for Physical layer. |
| Remote Wake-Up | The ability to power on a networked PC that is in standby or suspend mode using a wake-up event. |
| UDP | User datagram protocol. |
| UPD | Upload packet descriptor. |
| Upload | The process of transferring receive data from the NIC to system memory. |
| WOL | Wake on LAN (also known as Remote Wake-Up). |
| Word | A 16-bit wide quantity of data (2 bytes). |

**Register Bit Maps** The register descriptions in this technical reference include register bit maps. For example:




The first row of a bit map shows the bit numbers.

The second row of a bit map indicates the following information:

- Shaded areas indicate active register bits. The functions of these bits are described in the register descriptions.
- Unshaded areas in a register bit map indicate bits that disregard data written to them and return zeros when read. To ensure compatibility with future hardware, drivers should write zeros to these bits.
- Vertical lines mark the boundaries of fields of bits (for example, [12:0]).

# 2 ARCHITECTURE

This chapter describes the NIC system architectures and ASIC block diagrams, and summarizes the layout of the host registers and windows.

## 3C90x NICs Block Diagrams

The block diagrams for the 3C90x NICs are shown in Figure 1 through Figure 4. The NIC devices are described at the end of this chapter.

**Figure 1** 3C900-TPO System Architecture

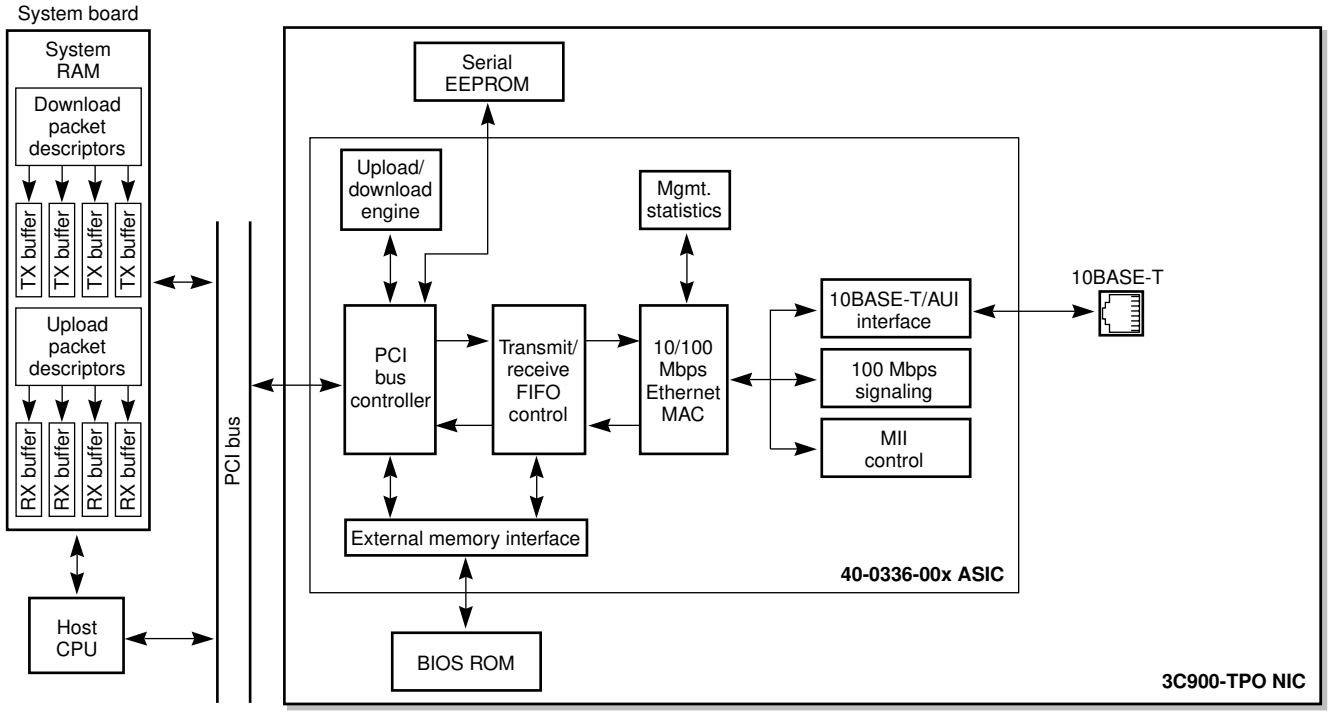


**Figure 2** 3C900-COMBO System Architecture

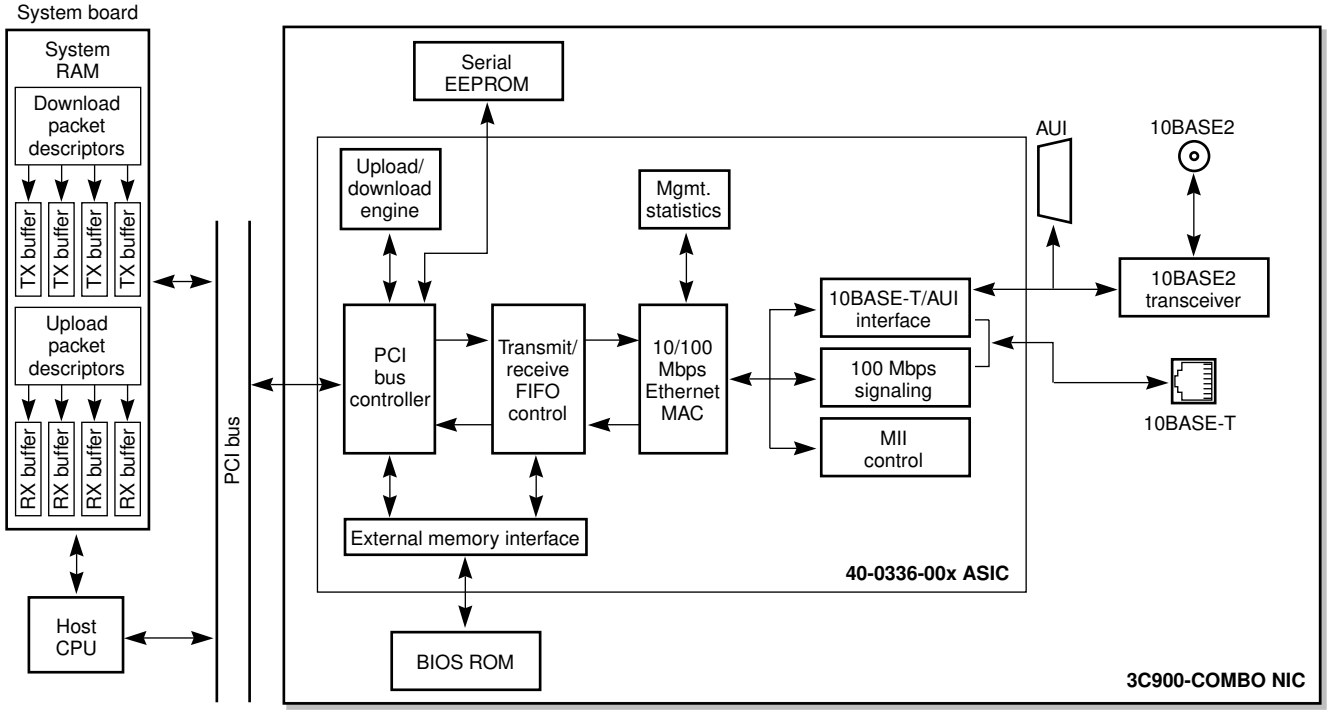



**Figure 3** 3C905-TX System Architecture

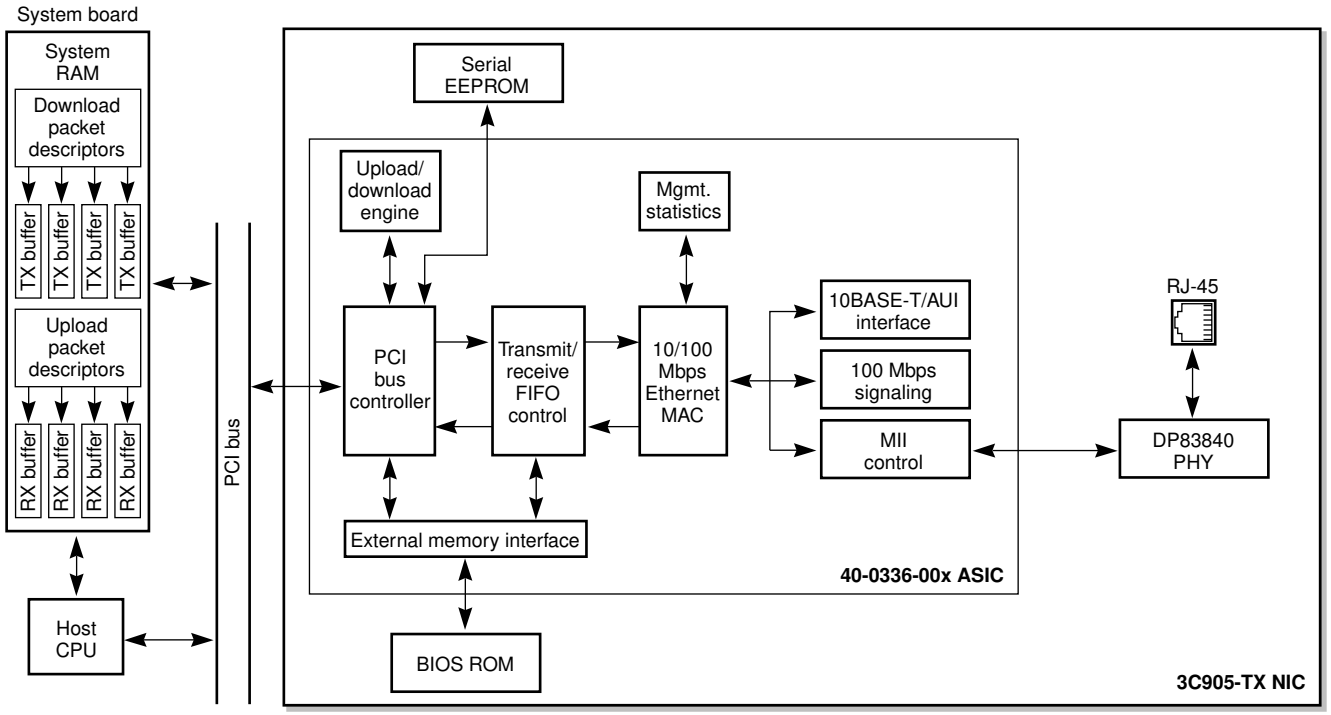
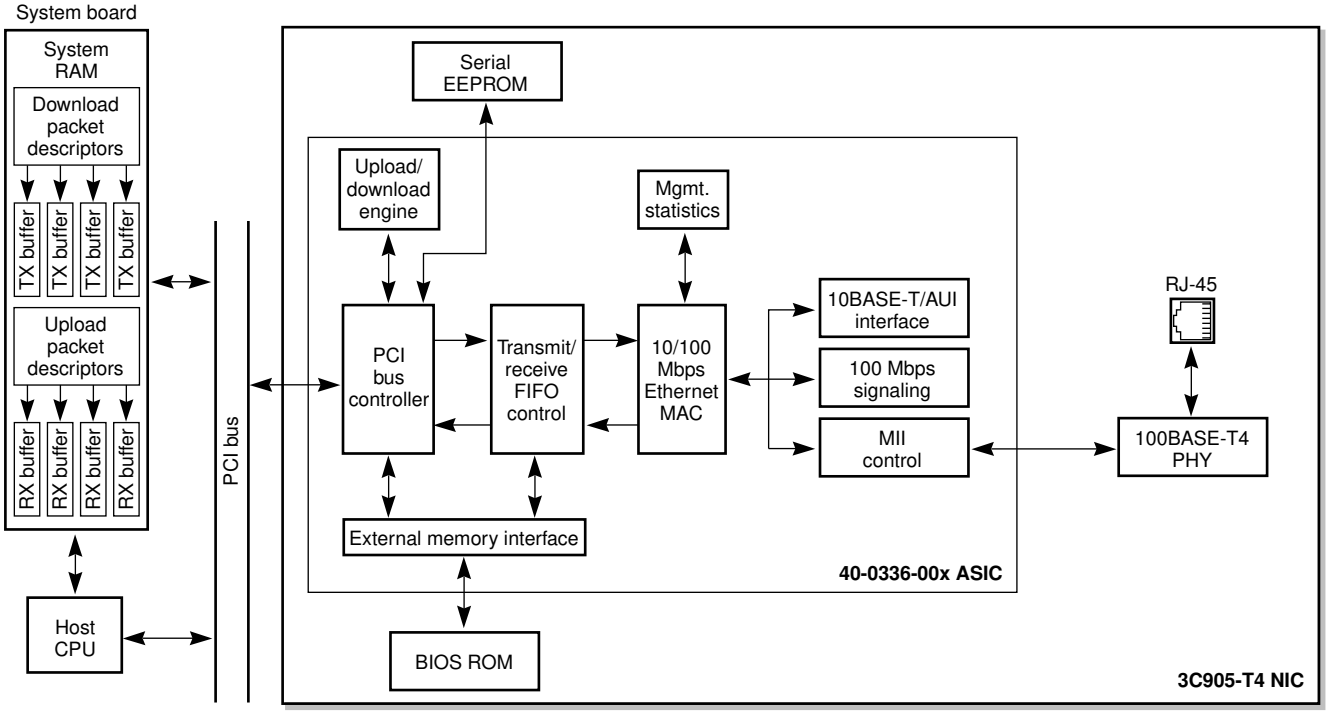


**Figure 4** 3C905-T4 System Architecture




## 3C90xB NICs Block Diagrams

The block diagrams for the 3C90xB NICs are shown in Figure 5 through Figure 12. The NIC devices are described at the end of this chapter.

**Figure 5** 3C900B-TPO System Architecture

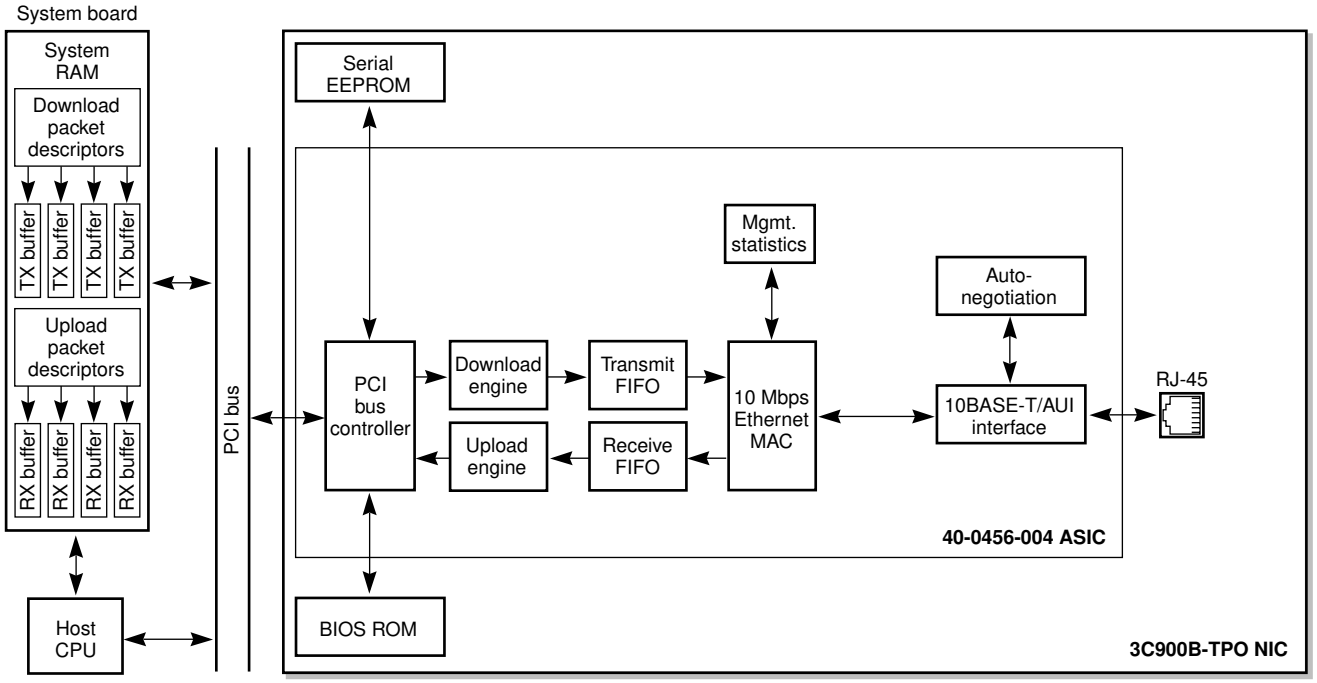


**Figure 6** 3C900B-TPC System Architecture

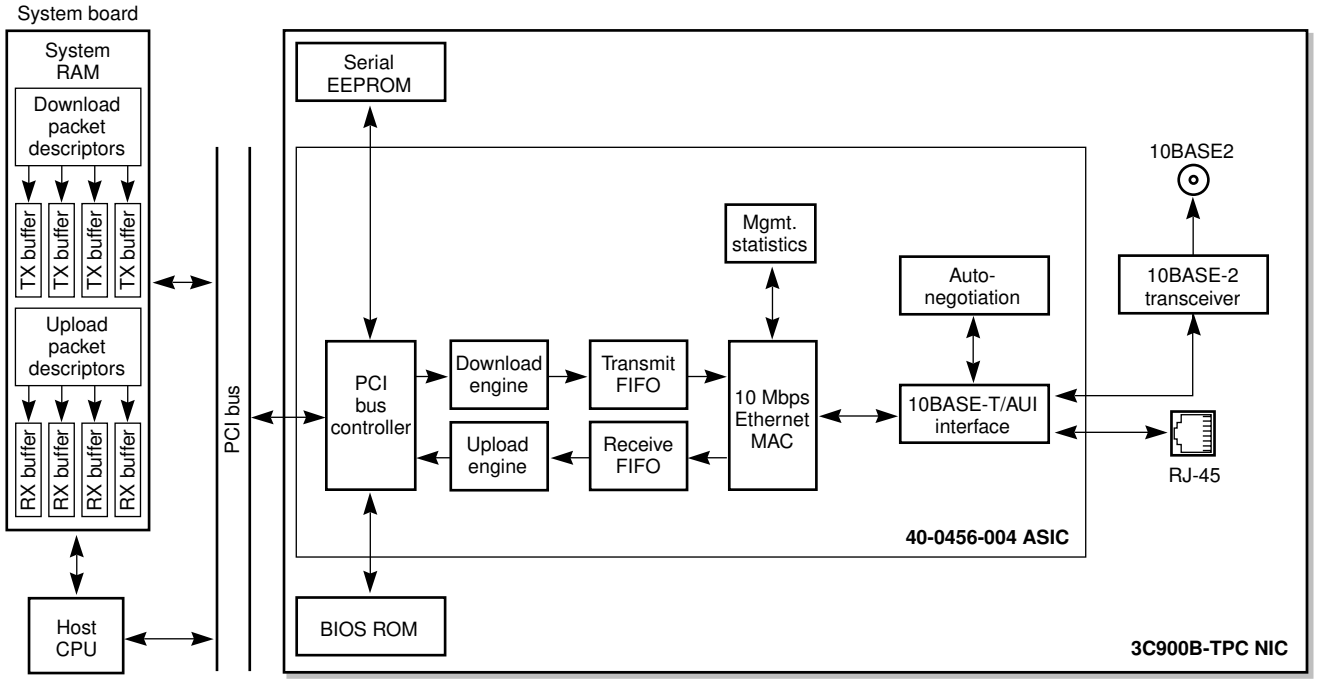



**Figure 7** 3C900B-COMBO System Architecture

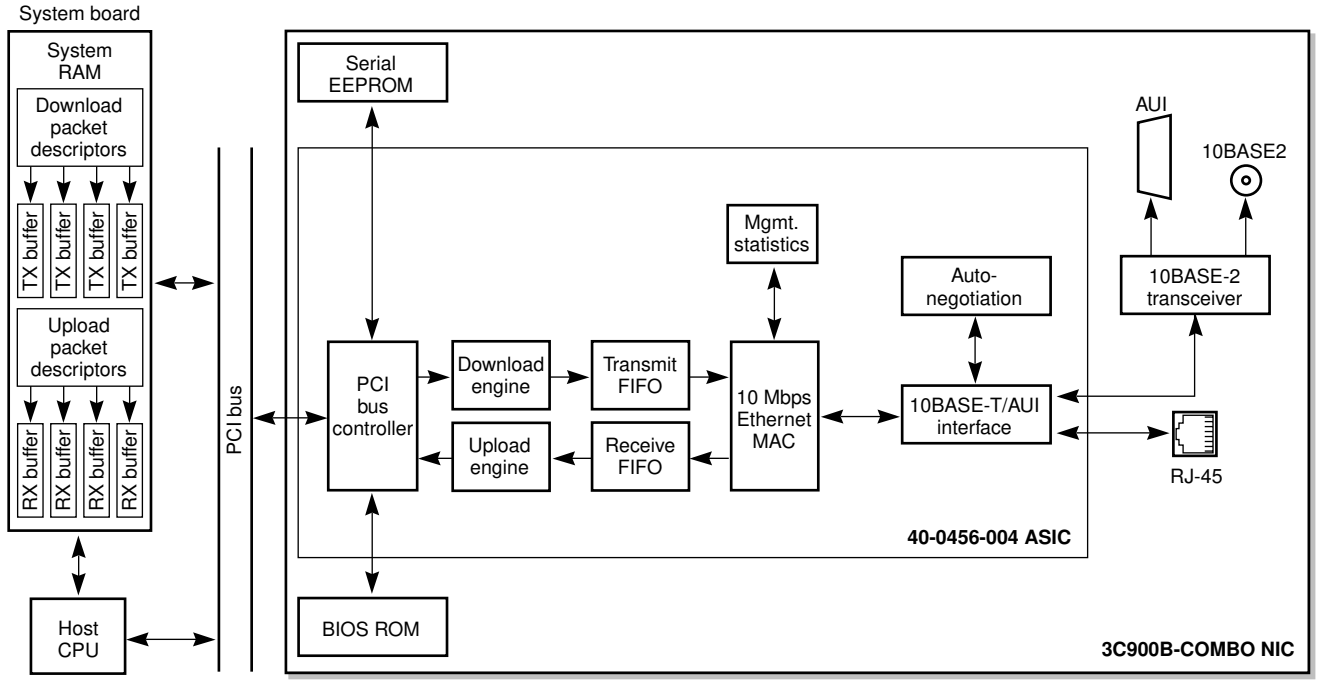


**Figure 8** 3C905B-TX with 40-0502-00x ASIC System Architecture

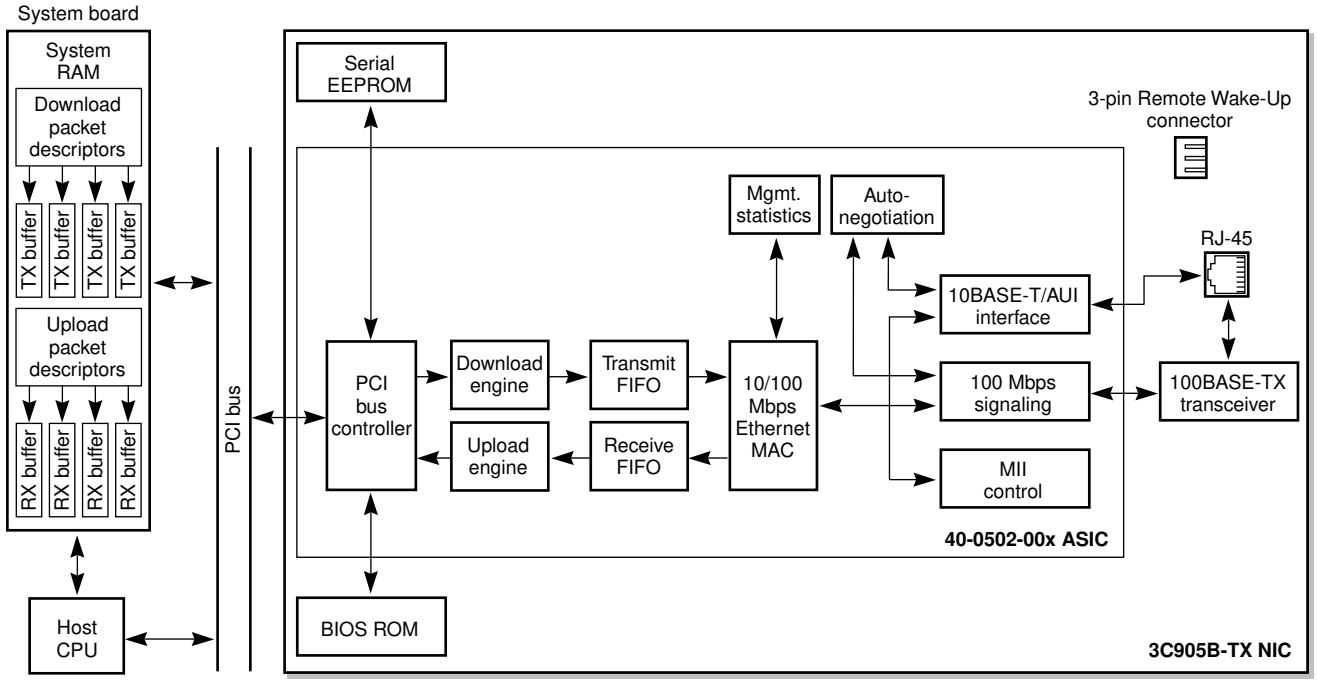



**Figure 9** 3C905B-TX with 40-0476-001 or 40-0483-00x ASIC System Architecture

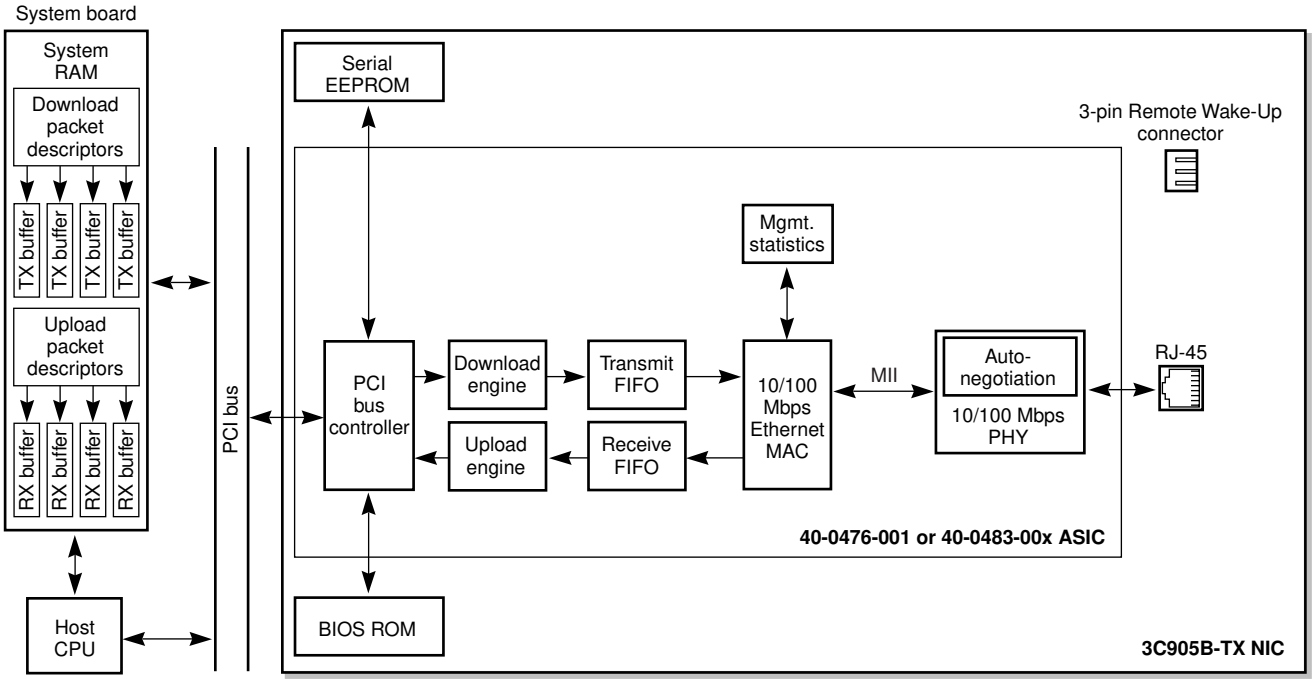


**Figure 10** 3C905B-TX-NM System Architecture

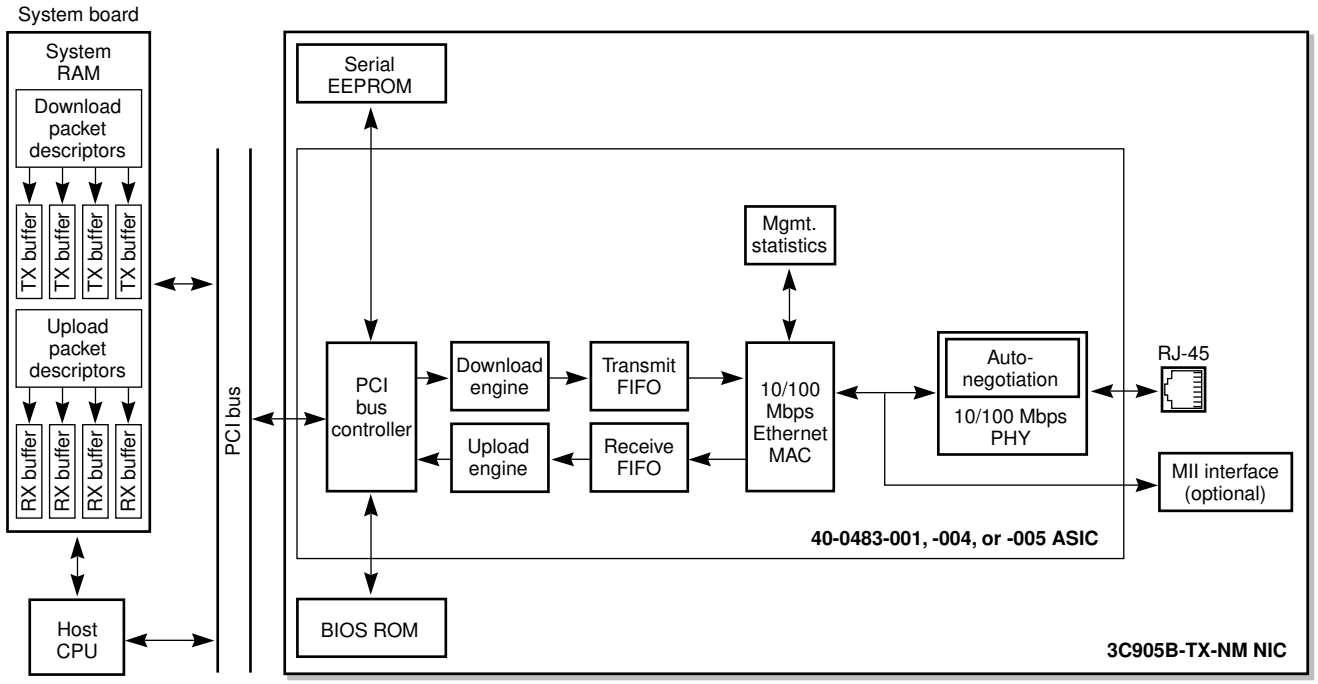



**Figure 11** 3C900B-FL System Architecture

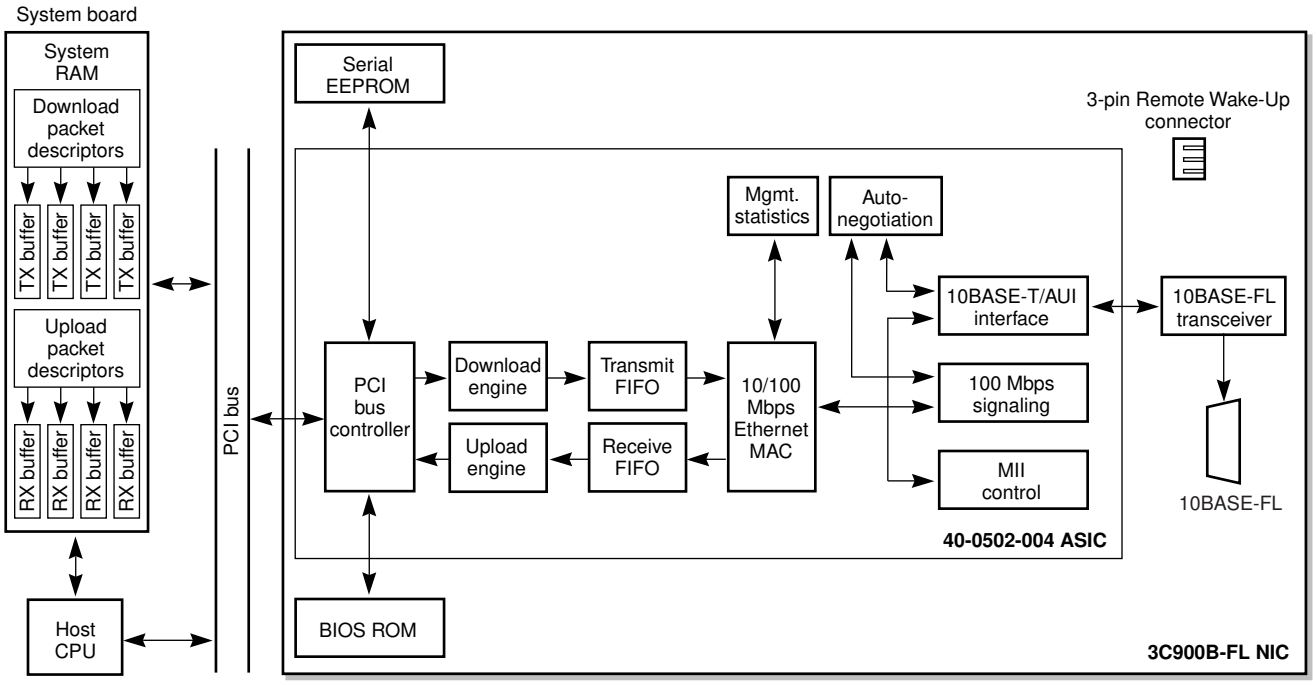
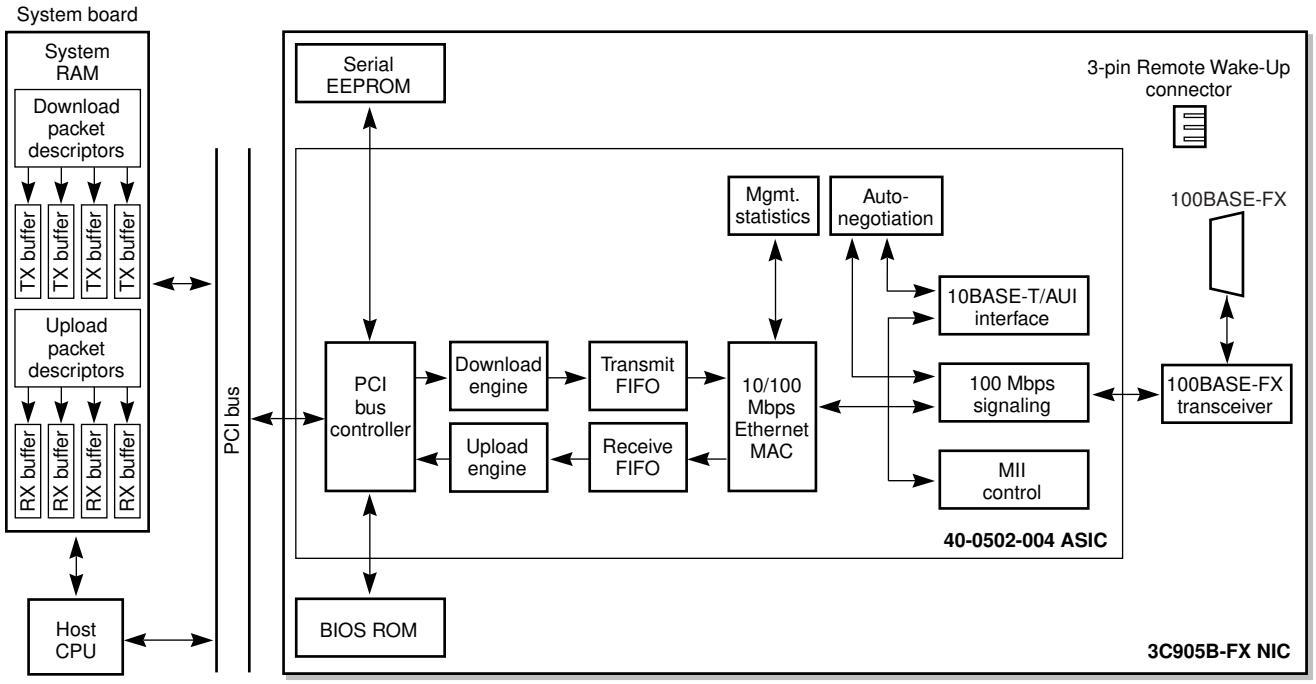


**Figure 12** 3C905B-FX System Architecture




## ASICs

The ASIC used by each NIC is listed in Table 3.

**Table 3** ASICs Summary

| NIC | ASIC Number(s) |
|---|---|
| **3C90x NICs** | |
| 3C900-TPO | 40-0336-000, -001, -002, -003, or -004 |
| 3C900-COMBO | 40-0336-000, -001, -002, -003, or -004 |
| 3C905-TX | 40-0336-000, -001, -002, -003, or -004 |
| 3C905-T4 | 40-0336-000, -001, -002, -003, or -004 |
| **3C90xB NICs** | |
| 3C900B-TPO | 40-0456-004 |
| 3C900B-TPC | 40-0456-004 |
| 3C900B-COMBO | 40-0456-004 |
| 3C905B-TX | 40-0502-001, -002, -003, or -004 |
| | 40-0476-001 |
| | 40-0483-001, -002, -004, or -005 |
| 3C905B-TX-NM | 40-0483-001, -004, or -005 |
| 3C900B-FL | 40-0502-004 |
| 3C905B-FX | 40-0502-004 |

*There are three different versions of the ASIC on the 3C905B-TX NIC. These three ASICs function identically except for the PHY portion, which is proprietary to each ASIC. As a result, the MII register layouts differ on the three ASICs. See Chapter 11 for information on the MII register layouts of each ASIC.*

### Hardware Identification

The ASIC on a 3C90x or 3C90xB NIC can be identified by the number that is inscribed on the ASIC. See Table 3 for a list of ASIC numbers.

### Software Identification

The ASIC on a 3C90xB NIC can be identified through software by viewing the chip/Vendor bit in the RevisionId register.

The following values in the chip/Vendor bit identify the ASIC on the NIC:

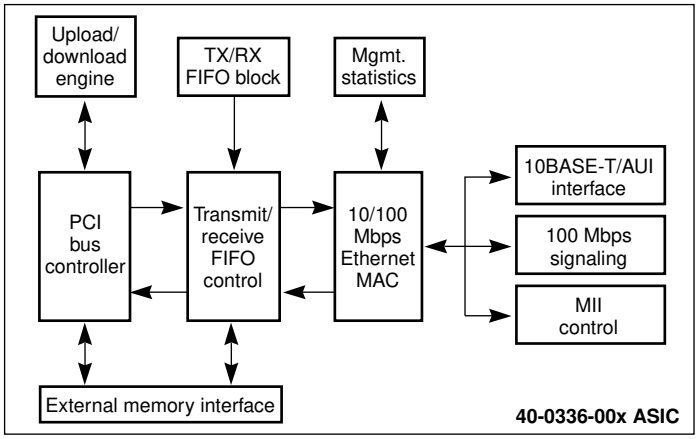
| chip/Vendor Bit Value | ASIC |
|---|---|
| 000 | 3C90xB NICs with the 40-0502-00x ASIC |
| 001 | 3C90xB NICs with the 40-0483-00x ASIC |
| 011 | 3C90xB NICs with the 40-0476-001 ASIC |

For more information on the RevisionId register, see "RevisionId" in Chapter 4.

### 3C90x NICs ASIC Diagram

The ASIC used on the 3C90x NICs is shown in Figure 13.

**Figure 13** 3C90x NICs ASIC Block Diagram




### 3C90xB NICs ASIC Diagrams

The ASICs used by the 3C90xB NICs are shown in Figure 14 through Figure 19.

**Figure 14** 3C900B NICs ASIC Block Diagram

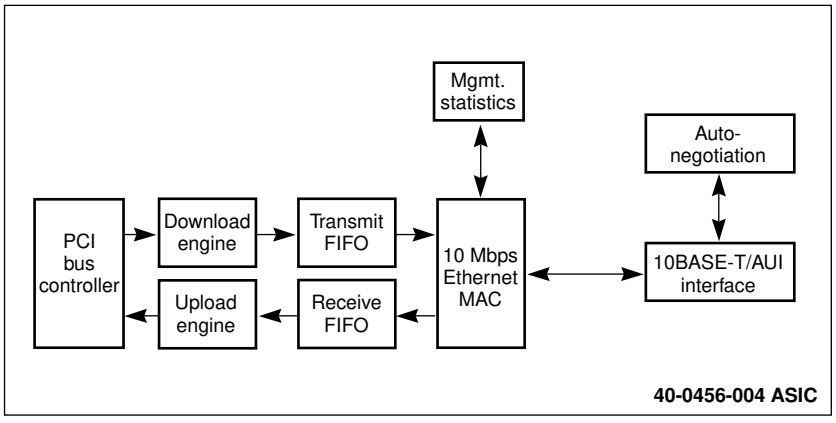


**Figure 15** 3C905B-TX NIC—40-0502-00x ASIC Block Diagram

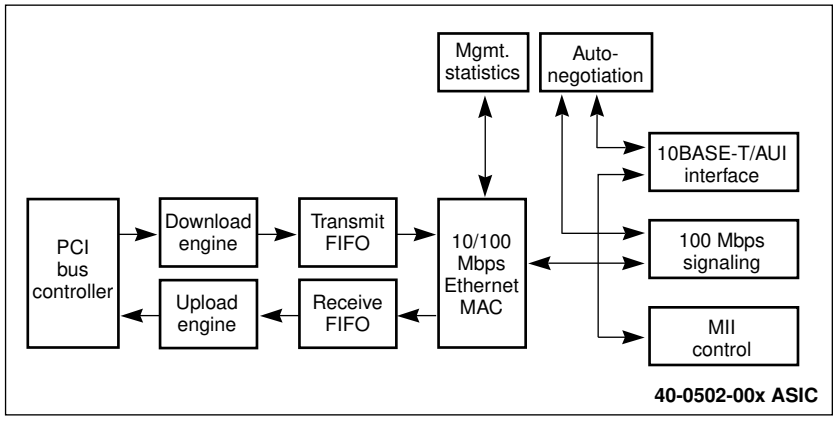



**Figure 16** 3C905B-TX NIC—40-0476-001 ASIC Block Diagram

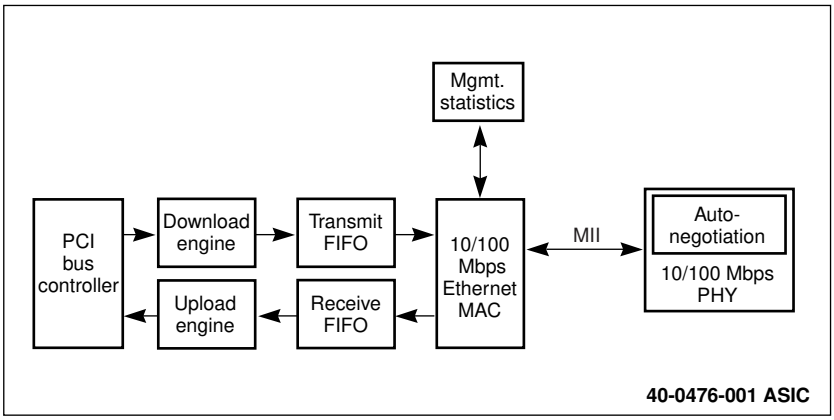



**Figure 17** 3C905B-TX NIC—40-0483-00x ASIC Block Diagram

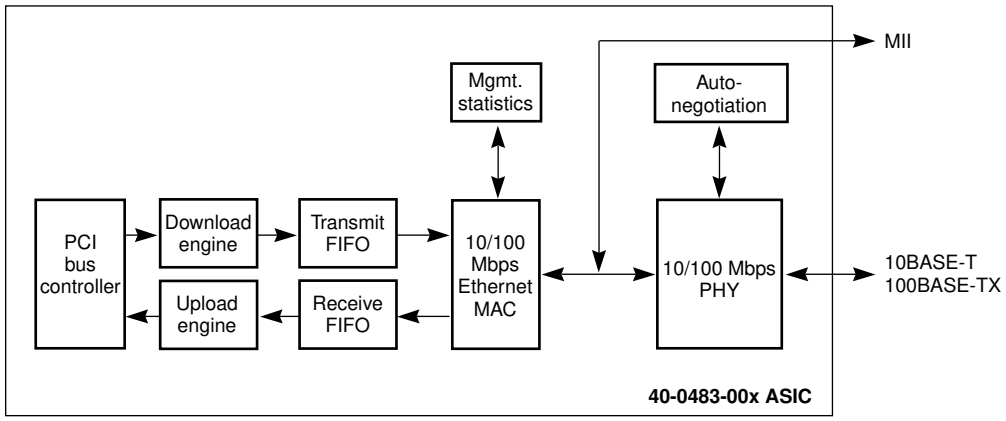


**Figure 18** 3C900B-FL NIC ASIC Block Diagram

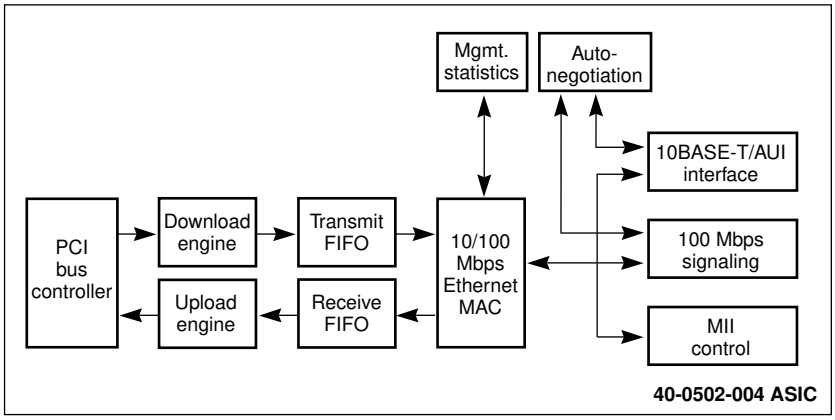
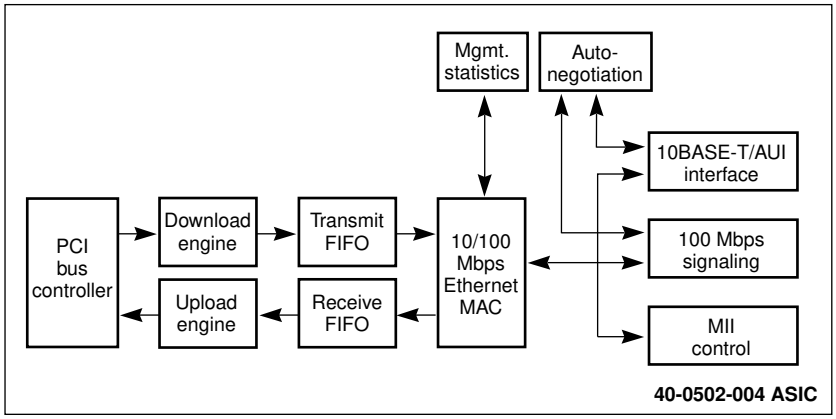



**Figure 19** 3C905B-FX NIC ASIC Block Diagram




## ASIC Block Descriptions

The devices associated with the ASICs are described below.

### PCI Bus Controller

This block implements the PCI interface functions (responding to PCI target cycles, generating and controlling PCI master cycles, and performing parity checking and generation).

The PCI bus controller logic also provides bus master services to the download and upload engines and provides the logic to control the BIOS ROM and serial EEPROM devices (shown in Figure 1 through Figure 12).

**Upload and Download Engines**

These blocks fetch the descriptors in the uplist and downlist and perform bus master data transfers by requesting PCI bus master burst service from the PCI bus controller block.

The upload engine removes receive data from the receive FIFO and supplies it to the PCI bus as it is required.

The download engine pipes transmit data from the PCI bus into the transmit FIFO.

**Transmit and Receive FIFOs**

These blocks are high-speed burst caches.

- On 3C90x NICs, this block contains 8 KB of buffer space, which can be partitioned for transmission and reception.
- On 3C90xB NICs, these blocks each contain 2 KB of data buffering and the logic required to manage the FIFOs.

**10/100 Mbps Ethernet MAC**

This block implements the IEEE 802.3 Media Access Control (MAC) function. It is responsible for the media access protocol, including deference, collision recovery and back off, receive packet filtering, and error detection. This block also provides information to the management statistics function.

**Management Statistics**

This block accumulates various network events statistics in hardware. Driver software reads these statistics periodically to maintain a network management information base (MIB).

**Auto-Negotiation**

On 3C90xB NICs that use the 40-0502-00x ASIC, this block provides the IEEE 802.3u auto-negotiation function.

On 3C90xB NICs that use the 40-0476-001 or 40-0483-00x ASIC, this block is integrated within the 10/100 Mbps PHY.

Auto-negotiation provides a means for the two devices in a link segment to communicate their signaling capabilities and automatically select the best mode. Auto-negotiation only manages twisted-pair–based signaling, so on the 3C90xB NICs, it covers only the 10BASE-T and 100BASE-TX ports. On 3C900B NICs, it covers 10BASE-T signaling.

**10BASE-T/AUI Interface**

This block provides the 10 Mbps encoder/decoder and transceiver functions to support 10BASE-T, AUI, and thin coax (10BASE2) media types. Only filtering magnetics are required off-chip to implement a complete 10BASE-T solution.

**100 Mbps Signaling**

This block provides the convergence sublayer and scrambling/descrambling functions required by the 100BASE-X specification.

- On 3C90x NICs, clock recovery and parallel-to-serial and serial-to-parallel conversions required to support the external 100BASE-TX transceiver chip are performed outside the ASIC.
- On 3C90xB NICs with the 40-0502-00x ASIC, this block includes the clock recovery and parallel-to-serial and serial-to-parallel conversions required to support the external 100BASE-TX transceiver chip.

On 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC, the 100 Mbps signaling function is implemented within the 10/100 Mbps PHY.

**10/100 Mbps PHY**

This block is on 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC. This block replaces the 10BASE-T/AUI, 100 Mbps signaling, and auto-negotiation functions found in earlier NICs.

**MII Control Logic**

This block contains the logic required to support the Media Independent Interface portion of the IEEE 802.3u standard. This allows the NIC chip to connect to any physical signaling chip that supports the MII standard.

On 3C905B-TX NICs with the 40-0476-001 or 40-0483-00x ASIC, this block is integrated within the 10/100 Mbps PHY.

## Other NIC Devices

The other devices associated with NIC operation are described below.

**BIOS ROM**

The optional BIOS ROM can contain code that is executed at system boot time.

- 3C90x NICs can contain up to 64 KB of code.
- 3C90xB NICs can contain up to 128 KB of code.

NICs generally come with a BIOS ROM socket that allows field installation and upgrade of the boot code. Support for Atmel PEROMs in the BIOS ROM socket allows the code to be updated without opening up the host computer.

**Serial EEPROM**

The 16-bit × 64-word serial EEPROM stores configuration information for the NIC, including PCI device ID, station address, and transceiver selection.

**External Media Transceivers**

The external media transceivers on the 3C90x and 3C90xB NICs include the following:

- 10BASE2
- DP83840 PHY
- 100BASE-T4 PHY
- 100BASE-TX
- 10BASE-FL
- 100BASE-FX

## Host Registers

This section shows the host register layouts for the 3C90x and 3C90xB NICs.

The NIC interacts with the host CPU through registers. The registers are mapped as follows:

- For 3C90x NICs, the registers are mapped into 64 bytes of the host CPU's I/O space.
- For 3C90xB NICs, the registers are mapped into 128 bytes of the host CPU's I/O space, memory space, or both. (Although registers are sometimes called "I/O registers," they may in fact be mapped and accessed in memory space.)

The first 16 bytes in the register space are a switchable window into one of eight register banks. The currently visible window is changed by issuing the SelectRegisterWindow command to the NIC. (The default window is zero.) The remaining bytes are a flat address decode and are always visible.

A register's location is specified by its offset from a base address that is defined in the IoBaseAddress PCI register, or, in the case of registers residing within a window, its window number and its offset within the window. For example, in 3C90xB NICs, the address to be used for I/O access of the BIOS ROM is held in the BiosRomAddr register in window 0, offset 7.

### Bit Widths of Register Accesses

In general, registers must be accessed as operands that are no wider than the bit width of the register. For example, although the BytesRcvdOk, UpperFramesOk, and FramesDeferred registers all appear in the double word at offset 8 in window 6, it is not legal to read all three registers with a single 32-bit I/O read instruction. Some registers cannot be accessed with cycles narrower than the register. Specific register access limitations are described in the register definitions in this technical reference.

### Command Register

Many of a driver's interactions with the NIC are performed using a command structure. Commands are codes, which sometimes include a parameter, that are written to the NIC to perform some action. For example, the RxEnable command causes the NIC transceiver to start accepting receive packets from the network.

Commands are written to the write-only Command register, which appears at offset e in every window. For details on the commands, see Chapter 10, "Command Register."

### Interrupt Status Register

The read-only IntStatus register shares the location offset e with the write-only Command register. A driver uses IntStatus to determine the sources of interrupts on the NIC. IntStatus also includes a bit that indicates when a command issued to the Command register is in the process of being executed.

The 3C90xB NICs have a special version of the IntStatus register, the IntStatusAuto register. Reading IntStatusAuto returns the value in IntStatus and causes some side effects that help optimize interrupt service routines.

### 3C90x NICs Register Layout

The host register layout and register window layout for the 3C90x NICs are shown in Table 4 and Table 5.

*Shaded areas indicate reserved spaces that are not implemented. Do not program in these spaces.*

**Table 4** 3C90x NICs Register Layout

| Byte 3 | Byte 2 | Byte 1 | Byte 0 | Offset |
|---|---|---|---|---|
| | | | | 3c |
| UpListPtr | | | | 38 |
| Countdown | | FreeTimer | | 34 |
| UpPktStatus | | | | 30 |
| TxFreeThresh | | | | 2c |
| | | | | 28 |
| DnListPtr | | | | 24 |
| DmaCtrl | | | | 20 |
| IntStatus/Command | | | | 1c |
| TxStatus | Timer | | | 18 |
| | | | | 14 |
| | | | | 10 |
| Register Windows 0 through 7 (See Table 5.) | | | | c |
| | | | | 8 |
| | | | | 4 |
| | | | | 0 |

**Table 5** 3C90x NICs Register Window Layout

| Byte 3 | Byte 2 | Byte 1 | Byte 0 | Offset | Window |
|---|---|---|---|---|---|
| IntStatus/Command | | | | c | 7 |
| TxStatus | Timer | | | 8 | |
| | | | | 4 | |
| | | | | 0 | |
| IntStatus/Command | | BytesXmittedOk | | c | 6 |
| BytesRcvdOk | | UpperFramesOk | FramesDeferred | 8 | |
| FramesRcvdOk | FramesXmittedOk | RxOverruns | LateCollisions | 4 | |
| SingleCollisions | MultipleCollisions | SqeErrors | CarrierLost | 0 | |
| IntStatus/Command | | IndicationEnable | | c | 5 |
| InterruptEnable | | | RxFilter | 8 | |
| RxEarlyThresh | | | | 4 | |
| TxAvailableThresh (not supported) | | TxStartThresh | | 0 | |
| IntStatus/Command | | UpperBytesOk | BadSSD | c | 4 |
| MediaStatus | | PhysicalMgmt | | 8 | |
| NetworkDiagnostic | | FifoDiagnostic | | 4 | |
| VcoDiagnostic (not supported) | | | | 0 | |
| IntStatus/Command | | TxFree | | c | 3 |
| RxFree | | ResetOptions | | 8 | |
| MacControl | | | | 4 | |
| InternalConfig | | | | 0 | |
| IntStatus/Command | | | | c | 2 |
| StationMask (Hi) | | StationMask (Mid) | | 8 | |
| StationMask (Lo) | | StationAddress (Hi) | | 4 | |
| StationAddress (Mid) | | StationAddress (Lo) | | 0 | |
| IntStatus/Command | | TxFree | | c | 1 |
| TxStatus | Timer | RxStatus | | 8 | |
| | | | RxError | 4 | |
| | | | | 0 | |
| IntStatus/Command | | EepromData | | c | 0 |
| EepromCommand | | | | 8 | |
| | | | | 4 | |
| | | | | 0 | |

**3C90xB NICs Register Layout**

The host register layouts and register window layouts for the 3C90xB NICs are shown in Table 6 and Table 7.

*Shaded areas indicate reserved spaces that are not implemented. Do not program in these spaces.*

**Table 6** 3C90xB NICs Register Layout

| Byte 3 | Byte 2 | Byte 1 | Byte 0 | Offset |
|---|---|---|---|---|
| | | PowerMgmtCtrl (write-only) | | 7c |
| UpMaxBurst | | DnMaxBurst | | 78 |
| | | DebugControl | | 74 |
| DebugData | | | | 70 |
| | | | | 6c |
| | | | | 68 |
| | | | | 64 |
| | | | | 60 |
| | | | | 5c |
| | | | | 58 |
| | | | | 54 |
| | | | | 50 |
| | | | | 4c |
| | | | | 48 |
| | | | | 44 |
| RealTimeCnt | | | | 40 |
| | UpBurstThresh | UpPoll | UpPriorityThresh | 3c |
| UpListPtr | | | | 38 |
| Countdown | | FreeTimer | | 34 |
| UpPktStatus | | | | 30 |
| | | DnPoll | DnPriorityThresh | 2c |
| | DnBurstThresh | | | 28 |
| DnListPtr | | | | 24 |
| DmaCtrl | | | | 20 |
| IntStatusAuto | | | | 1c |
| TxStatus | Timer | | TxPktId | 18 |
| | | | | 14 |
| | | | | 10 |
| Register Windows 0 through 7<br>(See Table 7.) | | | | c |
| | | | | 8 |
| | | | | 4 |
| | | | | 0 |

**Table 7** 3C90xB NICs Register Window Layout

| Byte 3 | Byte 2 | Byte 1 | Byte 0 | Offset | Window |
|---|---|---|---|---|---|
| IntStatus /Command | | PowerMgmtEvent | | c | 7 |
| | | | | 8 | |
| | | VlanEtherType | | 4 | |
| | | VlanMask | | 0 | |
| IntStatus /Command | | BytesXmittedOk | | c | 6 |
| BytesRcvdOk | | UpperFramesOk | FramesDeferred | 8 | |
| FramesRcvdOk | FramesXmittedOk | RxOverruns | LateCollisions | 4 | |
| SingleCollisions | MultipleCollisions | SqeErrors | CarrierLost | 0 | |
| IntStatus /Command | | IndicationEnable | | c | 5 |
| InterruptEnable | | TxReclaimThresh | RxFilter | 8 | |
| RxEarlyThresh | | | | 4 | |
| | | TxStartThresh | | 0 | |
| IntStatus /Command | | UpperBytesOk | BadSSD | c | 4 |
| MediaStatus | | PhysicalMgmt | | 8 | |
| NetworkDiagnostic | | FifoDiagnostic | | 4 | |
| VcoDiagnostic (not supported) | | | | 0 | |
| IntStatus /Command | | TxFree | | c | 3 |
| RxFree | | MediaOptions | | 8 | |
| MacControl | | MaxPktSize | | 4 | |
| InternalConfig | | | | 0 | |
| IntStatus /Command | | ResetOptions | | c | 2 |
| StationMask (Hi) | | StationMask (Mid) | | 8 | |
| StationMask (Lo) | | StationAddress (Hi) | | 4 | |
| StationAddress (Mid) | | StationAddress (Lo) | | 0 | |
| IntStatus /Command | | | | c | 1 |
| | | | | 8 | |
| | | | | 4 | |
| | | | | 0 | |
| IntStatus /Command | | EepromData | | c | 0 |
| EepromCommand | | | BiosRomData | 8 | |
| BiosRomAddr | | | | 4 | |
| | | | | 0 | |

# 3 OPERATION

This chapter summarizes NIC operational characteristics.

## Data Structure Lists

To move data between the host and the NIC, drivers set up data structures in system RAM to specify the buffers to be used for packet data movement. These data structures, called descriptors, are linked together in system memory to form lists.

All packet data is moved across the NIC PCI bus by bus master operations. The NIC also uses bus master operations to read descriptor information out of system RAM and to write status back into the descriptors.

Movement of a transmit packet to the NIC is called a download. The list of download packet descriptors (DPDs) is called the *downlist*. Similarly, a receive packet movement is called an upload, and the list of upload packet descriptors (UPDs) is the *uplist*.

The device driver creates and maintains the uplist and the downlist. It starts the download process by writing the address of the first download descriptor in the downlist to the DnListPtr register. Uploads are started by writing the first upload descriptor address to the UpListPtr register. The device driver also accesses NIC registers for initialization, interrupt handling, statistics collection, and error handling.

For details on data structure lists, see Chapter 6 and Chapter 7.

## PCI Bus Master Operation

This section describes aspects of bus master operation that can be controlled by software. For information about PCI configuration, see "PCI Configuration Registers" in Chapter 4.

### PCI Memory Commands

The 3C90x NICs support two PCI memory commands (see Table 8). The 3C90xB NICs support all PCI memory commands (see Table 9).

**Table 8** 3C90x NICs PCI Memory Commands

| Command | Description |
|---|---|
| MW | Memory Write |
| MRM | Memory Read Multiple |

**Table 9** 3C90xB NICs PCI Memory Commands

| Command | Description |
|---|---|
| MW | Memory Write |
| MWI | Memory Write Invalidate |
| MR | Memory Read |
| MRL | Memory Read Line |
| MRM | Memory Read Multiple |

MR is used for all fetches of descriptor information. For reads of transmit packet data, MR, MRL, or MRM is used, depending upon the remaining number of bytes in the fragment, the amount of free space in the transmit FIFO, and whether the upload engine is requesting a bus master operation.

MW is used for all descriptor writes. Writes of receive packet data use either MW or MWI, depending upon the remaining number of bytes in the fragment, the amount of packet data in the receive FIFO, and whether the download engine is requesting a bus master operation.

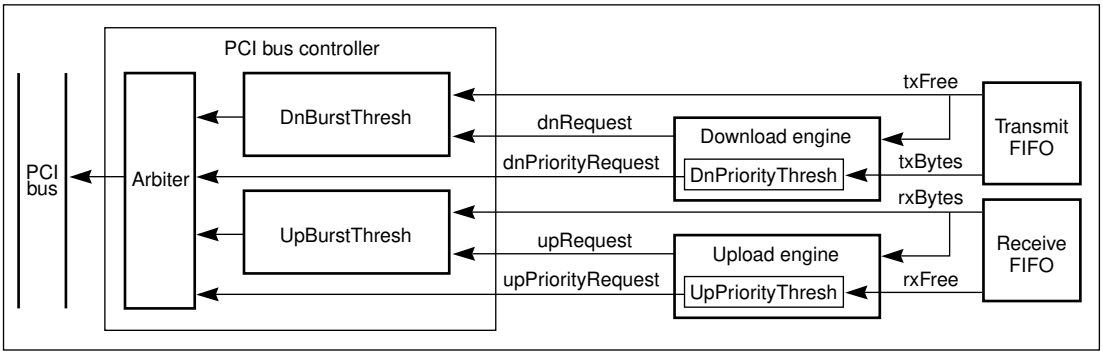
The 3C90xB NICs have three configuration bits that control the use of PCI memory commands:

- The MWIEnable bit in the PciCommand configuration register allows the system to enable or disable the NIC's use of MWI.
- The defeatMWI bit in the DmaCtrl register can disable the NIC's use of MWI. By default, MWI is enabled.
- The defeatMRL bit in the DmaCtrl register can disable the NIC's use of MRL. By default, MRL is enabled.

## PCI Bus Request Control

The 3C90xB NICs provide a set of registers that control PCI burst behavior. These registers allow trade-offs to be made between PCI bus efficiency and underrun/overrun frequency. Figure 20 illustrates the bus request structure.

**Figure 20** 3C90xB NICs Bus Request Structure




Arbitration logic (the arbiter) within the PCI bus controller block accepts bus requests from the download and upload engines.

### Download

The download engine monitors the amount of free space in the transmit FIFO. When there are at least 16 bytes of free space and a fragment available for download, the download engine uses the dnRequest bit to make a standard bus request. The DnBurstThresh logic register qualifies dnRequest. When the amount of free space in the FIFO is greater than the value in the DnBurstThresh register, a download request is passed on to the arbiter. The purpose of DnBurstThresh is to delay the bus request until there is enough free space in the FIFO for a long, efficient burst.

The download engine also has a way to make an emergency bus request. When the number of used bytes in the FIFO drops below the value in the DnPriorityThresh register, indicating that the FIFO is approaching an underrun condition, the dnPriorityRequest bit makes a priority request. This request is not subject to the DnBurstThresh constraint; when the FIFO is close to underrun, burst efficiency is sacrificed in favor of requesting the bus as quickly as possible.

### Upload

The upload mechanism is similar to download. The upload engine monitors the number of bytes in the receive FIFO. When there are enough bytes to make the packet visible and a buffer is available for upload, the upload engine uses the upRequest bit to make a standard bus request. The UpBurstThresh logic register qualifies the upRequest bit. When the number of bytes in the FIFO is greater than the value in the UpBurstThresh register, an upload request is passed on to the arbiter. Priority requests prevent receive overruns. The upPriorityRequest bit is asserted when the free space in the receive FIFO falls below the value in the UpPriorityThresh register.

The arbiter services the four requests in this fixed priority order:

1. upPriorityRequest
2. dnPriorityRequest
3. upRequest
4. dnRequest

## IEEE 802.3x Flow Control

The 3C90xB NICs support IEEE 802.3x flow control.

The IEEE 802.3x full-duplex supplement to the IEEE 802.3 standard specifies a MAC control sublayer that allows real-time control over the MAC sublayer. A new MAC control frame format provides station-to-station communication at the MAC control level.

A variety of MAC control commands is possible, but the 3C90xB NICs interpret only the PAUSE operation. The PAUSE command can be used to implement flow control. It allows one station to instruct another to inhibit transmission of data frames for a specified period.

The PAUSE control frame format appears in Figure 31 in Appendix C.

Whenever the flowControlEnable bit in the MacControl register is one, the NIC compares the destination address with 01:80:C2:00:00:01 and the type field with 88:08 in all incoming packets. If both fields match, the NIC further checks for the PAUSE opcode (00:01) in the MAC Control Opcode field. If this is found, the NIC inhibits transmission of all data frames (but *not* MAC control frames) for the time specified in the 2-byte pause_time field. The pause_time value is specified in terms of the number of slot times for the configured data rate:

- At 10 Mbps, one slot time is 51.2 μs.
- At 100 Mbps, one slot time is 5.12 μs.

The pause_time value is received from the medium most-significant byte first.

When the flowControlEnable bit is zero, MAC control packets are subject to the same receive filtering conditions as normal packets; the NIC stores a packet that satisfies one or more of the receive filters and makes the packet available for upload to the host.

## VLAN Support

The 3C90xB NICs include support for both IEEE 802.1Q VLANs and the 3Com proprietary VLAN Tagging (VLT) scheme.

For IEEE 802.1Q and VLT frame formats, see Appendix C.

### IEEE 802.1Q VLANs

IEEE 802.1Q frames have four extra bytes (over and above the usual IEEE 802.3 frame format) that are inserted between the source address and the Length/type field. Two of the bytes are a special type (TPID). The two other bytes contain a 12-bit VLAN ID number, three bits of priority, and a token ring encapsulation bit.

A driver can use the MaxPktSize register to set the size at which packets are flagged as oversize, allowing the NIC to accommodate the increased IEEE 802.1Q packet size. A driver using IEEE 802.1Q formats should increase the value in MaxPktSize by four over the default.

The NIC interprets IEEE 802.1Q frames only for the purpose of properly performing TCP/IP checksumming. The VlanEtherType register is programmed with the value of TPID (at the time of writing, this value had not been defined). When the checksumming logic encounters a packet in which the thirteenth and fourteenth bytes match VlanEtherType, it recognizes the packet as IEEE 802.1Q and skips the thirteenth through sixteenth bytes in its checksum calculation.

### 3Com VLT

VLT frames have four extra bytes inserted before the destination address. Within these four bytes is a 4-bit VLAN ID that indicates to which VLAN the packet is directed.

On 3C90xB NICs, VLT is enabled globally with the vltEnable bit in the MacControl register. When vltEnable is set, every packet received by the NIC is expected to have the four VLT bytes. The VLT bytes are not removed automatically by the hardware; the driver must remove them.

To allow the NIC to accommodate the increased size of VLT packets, the MaxPktSize register allows the driver to set the size at which packets are flagged as oversize. When vltEnable is set, a driver should also increase the value in MaxPktSize by four over the default.

The 3C90xB NICs can receive packets addressed to multiple VLANs. By default, when VLT is enabled, the NIC receives all VLT packets, regardless of the contents of the ID field. The VlanMask register allows individual ID tags to be masked, causing the NIC to discard packets containing those ID values.

The 3C90xB NICs also interpret VLT frames for the purpose of TCP/IP checksumming. When the vltEnable bit is set, the checksumming logic skips the first through fourth bytes in every packet during its checksum calculation.

## Power Management

The 3C90xB NICs support power management directed by the operating system, in accordance with the *Advanced Configuration and Power Management (ACPI) Specification*. The following paragraphs describe the power management features.

### Power Management Registers

Power management registers are in the PCI configuration space, as defined by the *PCI Bus Power Management Interface Specification, Revision 1.0*.

The PowerMgmtCap register supplies the system with information about the NIC's power management support and capabilities. The PowerMgmtCtrl register allows system or driver software to read the NIC's power management status and set the NIC's power state.

### Power States

Table 10 defines the supported power states. The current power state is determined by the powerState field in the PowerMgmtCtrl register.

**Table 10** 3C90xB NICs Power States

| State | powerState Value | Description |
|---|---|---|
| $D0_{uninitialized}$ | 0 | This state is a result of a hardware reset, or of a transition from $D3_{hot}$ to D0. This state is the same as $D0_{active}$ except that the PCI configuration registers are uninitialized. In this state, the NIC responds to PCI configuration cycles only. |
| $D0_{active}$ | 0 | This is the normal operational power state for the 3C90xB NICs.<br>In this state, the PCI configuration registers have been initialized by the system, including the ioSpace, memorySpace, and busMaster bits in the PciCommand register, so the NIC is able to respond to PCI I/O, memory and configuration cycles and can operate as a PCI master.<br>The 3C90xB NICs cannot signal wake-up (assert the PME# signal on the PCI bus) from the D0 state. |
| D1 | 1 | This is a "light-sleep" state, which allows transition back to D0 with no delay. Support for D1 is determined by the d1Support bit in the PciParm word in EEPROM.<br>In this state, the NIC responds to PCI configuration accesses, allowing the system to change the power state, but it does not respond to any PCI I/O or memory accesses.<br>The NIC's function in the D1 state is to recognize wake-up events and link state events and pass them on to the system by asserting the PME# signal on the PCI bus. |

(1 of 2)

**Table 10** 3C90xB NICs Power States (continued)

| State | powerState Value | Description |
|---|---|---|
| D2 | 2 | This is a partial power-down state that allows a faster transition back to D0 than is possible from the D3 state. Support for D2 is determined by the d2Support bit in the PciParm word in EEPROM.<br>Like the D1 state, the NIC in the D2 state responds to PCI configuration accesses, allowing the system to change the power state, but it does not respond to any PCI I/O or memory accesses.<br>Similarly, the function of the NIC in the D2 state is to recognize wake-up events and link state events and pass them on to the system by asserting the PME# signal on the PCI bus. |
| $D3_{hot}$ | 3 | This is the full power-down state for the 3C90xB NICs. In $D3_{hot}$, the NIC loses all PCI configuration information except for the value in the powerState bit.<br>In this state, the NIC responds to PCI configuration accesses, to allow the system to change the power state back to $D0_{uninitialized}$, but it does not respond to any PCI I/O or memory accesses.<br>The NIC's function in the $D3_{hot}$ state is to recognize wake-up events and link state events and pass them on to the system by asserting the PME# signal on the PCI bus. |
| $D3_{cold}$ | N/A | This is the power-off state for the 3C90xB NICs. The NIC has no function in this state. When power is restored, the system guarantees the assertion of hardware reset, which puts the NIC into the $D0_{uninitialized}$ state. |

(2 of 2)

## Remote Wake-Up

The 3C905B-TX, 3C900B-FL, and 3C905B-FX NICs support Remote Wake-Up, the ability to power on a networked PC that is in standby or suspend mode using a wake-up event.

A wake-up event is generated as a result of one of the following:

- Wake-up packet reception
- Magic Packet reception
- Change in link state

The PowerMgmtEvent register gives the driver control over which of these events is passed to the system. Wake-up events are signaled over the PCI bus using the PME# pin.

*The 3C90x, 3C900B, and 3C905B-TX-NM NICs do not support Remote Wake-Up.*

## Wake-up Packets

The 3C905B-TX, 3C900B-FL, and 3C905B-FX NICs can signal wake-up when the NIC receives an "interesting" packet from another station. Driver software defines interesting packets by downloading a set of frame patterns to the transmit FIFO before placing the NIC in a power-down state. Once the NIC is powered down, it compares receive packets with the frame patterns. Wake-up is signaled when a packet is received that matches a frame pattern and also passes the filter set in the RxFilter register.

A signature-matching technique allows the NIC to recognize wake-up packets. The frame patterns specify which bytes in the incoming packets are to be examined. A CRC is calculated over these bytes and compared with a CRC value supplied in the frame pattern. This matching technique may result in false wake-ups being reported to the system.

Packet wake-up is controlled by the wakeupPktEnable bit in the PowerMgmtEvent register. The wakeupPktEnable bit has no effect when the NIC is in the D0 power state; wake-up can only take place in the D1, D2, or D3 states. When the NIC detects a wake-up packet, it signals a wake-up event on PME# (if PME# assertion is enabled), and sets the wakeupPktEvent bit in PowerMgmtEvent.

### Downloading Wake-up Frame Patterns

Drivers download frame patterns to the transmit FIFO in a single "pseudo packet" as follows:

1. Issue a TxReset command (to reset the FIFO pointers and prevent transmission).
2. Prepare a DPD that points to a single data buffer.

   The buffer contains one or more frame patterns, placed contiguously. The transmit FIFO size limits the number of frame patterns. The DnFragLen DPD entry must exactly equal the sum of the frame pattern bytes.
3. Set the rndupDefeat bit in the Frame Start Header DPD entry to prevent rounding up of the packet size.
4. Write the DPD address to the DnListPtr register to download the packet.

### Wake-up Frame Patterns

Each wake-up frame pattern contains the following:

- One or more byte offset/byte count pairs—The byte offset indicates the number of packet bytes to be skipped in order to reach the next group of bytes to be included in the CRC calculation. The byte count indicates the number of bytes in the next group to be included in the CRC calculation.
- End-of-pattern symbol—This byte value (00) indicates the end of the pattern for that wake-up frame.
- Four-byte CRC value—This CRC value uses the same polynomial as the Ethernet MAC CRC.

The frame patterns are encoded as follows: The byte offset/byte count values are contained in a single byte. Bits [7:4] contain the byte offset value, and bits [3:0] contain the byte count. The byte offset and byte count can take on a value of 0 to 14d. A byte offset or byte count value of 15d indicates that it has an extended value: this value occupies eight bits and is contained in the next pattern byte. If both the byte offset and the byte count values are 15d, the next byte is the extended byte offset, and the byte after that is the extended byte count.

Offset/count bytes occur in the pattern until terminated by a zero byte, which indicates the end of pattern for that wake-up frame. Following the end of pattern are four bytes of CRC value. If there is another wake-up frame pattern, then it immediately follows the CRC value.

As an example, the following is the pattern to be downloaded into the transmit FIFO for the ARP wake-up frame shown in Appendix A of the *Network Device Class Specification*:

```
c2      // byte offset = c, byte count = 2
71      // byte offset = 7, byte count = 1
f4      // byte offset = extended, byte count = 4
10      // byte offset = 10h
00      // end of pattern
f3      // first byte of CRC
19      // second byte of CRC
08      // third byte of CRC
d7      // fourth byte of CRC
```

**Magic Packet Technology**

The 3C905B-TX, 3C900B-FL, and 3C905B-FX NICs can signal wake-up when the NIC receives a Magic Packet frame from another station.

The Magic Packet technology was developed by Advanced Micro Devices to allow remote wake-up of a sleeping station on a network. The technology involves sending a special packet to the sleeping station. Once a station has been placed in Magic Packet mode and put to sleep, it scans all incoming packets addressed to it for a specific data sequence.

The data sequence consists of 16 duplications of the Ethernet MAC address of the station, with no breaks or interruptions. This sequence can be located anywhere within the packet, but must be preceded by a synchronization stream. The synchronization stream is defined as six bytes of FFh.

The device also accepts a broadcast frame as long as the 16 duplications of the MAC address match the address of the machine to be awakened. If the MAC address for a particular node on the network was 11:22:33:44:55:66, then the LAN controller would be scanning for the following data sequence:

Destination_Address Source_Address {Miscellaneous} FF FF FF FF FF FF 11 22 33 44 55 66 11 22 33 44 55 66 11 22 33 44 55 66 11 22 33 44 55 66 11 22 33 44 55 66 11 22 33 44 55 66 11 22 33 44 55 66 11 22 33 44 55 66 11 22 33 44 55 66 11 22 33 44 55 66 11 22 33 44 55 66 11 22 33 44 55 66 {Miscellaneous} CRC

Magic Packet wake-up is controlled by the magicPktEnable bit in the PowerMgmtEvent register. The magicPktEnable bit has no effect when the NIC is in the D0 power state; wake-up can only take place in the D1, D2, or D3 states.

For the NIC to receive and recognize a Magic Packet frame, the packet must also pass the filter criteria set in the RxFilter register. When the NIC detects a Magic Packet frame, it signals a wake-up event on PME# (if PME# assertion is enabled), and sets the magicPktEvent bit in PowerMgmtEvent.

### Change of Link State

The 3C905B-TX, 3C900B-FL, and 3C905B-FX NICs can signal a wake-up event when a change in the network link state (either from LINK_OK to LINK_FAIL, or vice versa) is detected.

Link state wake-up is controlled by the linkEventEnable bit in the PowerMgmtEvent register. At the time linkEventEnable is set by software, the NIC samples the current link state. It then waits for the link state to change. If the link state changes before the NIC is returned to state D0 or linkEventEnable is cleared, the linkEvent bit is set in PowerMgmtEvent, and (if it is enabled) the PME# signal is asserted.

### Programming Remote Wake-Up Events

This section describes the sequences involved in programming the 3C905B-TX, 3C900B-FL, or 3C905B-FX NIC for Remote Wake-Up events.

#### Power Down

While in the operating state (D0), the device driver is notified by the operating system that a power state change is imminent. The device driver prepares for power down with these steps:

1. Halt the download process: stall the download engine; wait for any download in progress to finish; wait for any transmissions to end; issue a TxReset command to reset the FIFO pointers.
2. Perform uploads for any receive packets remaining in FIFO; stall the upload engine (packets potentially keep filling FIFO between now and when the operating system powers down the NIC. Once the power down state is entered and wake-up packet scanning is enabled, these packets are also scanned and potentially wake up the system immediately).
3. Clear the InterruptEnable register so that no interrupts occur before the powerState bit in the PowerMgmtCtrl register is changed.
4. Save any NIC volatile state to system memory (system memory is restored after a power down). Examples of volatile state are the pending power state and the hash filter settings.
5. Download wake-up frame patterns to the transmit FIFO (if wake-up packets are to be enabled).
6. Program the PowerMgmtEvent register to enable desired wake-up events.
7. Return to the operating system, indicating that the NIC is ready to be powered down. Ensure that the RxFilter register is programmed to receive the appropriate wake-up and Magic Packet frames.
8. The operating system eventually writes to the NIC's PowerMgmtCtrl register, placing the NIC in one of the power down states, and enabling PME# assertion.
9. The NIC scans packets for enabled wake-up events.

**Wake-Up**

1 When a desired wake-up event occurs, the NIC sets the appropriate event bit in the PowerMgmtEvent register, sets the pmeStatus bit in the PowerMgmtCtrl register, and asserts the PME# pin.

   Alternatively, some other device in the system could recognize a wake-up event, such as the user pushing the soft power button.

2 The system responds to PME# by scanning the power management configuration registers of all NICs, looking for the device that asserted PME#. If the NIC signaled the wake-up, the system finds pmeStatus set in the NIC's PowerMgmtCtrl register.

3 If the NIC signaled wake-up, the operating system clears the pmeEn bit in the PowerMgmtCtrl register, causing PME# to be deasserted.

4 The operating system raises the power state (probably to D0) by writing the powerState bit in the PowerMgmtCtrl register. If the NIC was previously in the D3 state, PCI configuration is lost and is restored by the operating system.

5 The operating system calls the NIC's device driver to inform it of the power state transition, and possibly to determine the nature of the wake-up event.

6 The device driver issues a TxReset command to clear any wake-up patterns out of the transmit FIFO (if this is not done, the patterns are treated as packets and transmitted once the transmitter is enabled).

7 The device driver reads the PowerMgmtEvent register to determine the wake-up event, and if requested, passes it back to the system.

8 The device driver restores any volatile state that was saved in the power down sequence.

9 The device driver reenables interrupts by programming the InterruptEnable register.

10 The device driver restores the uplist (and any other data structures required for operation). Any wake-up packet in the receive FIFO is uploaded and passed to the system or protocol.

*The system must be enabled in order to accept power management event wake-up signals in the BIOS.*

## TCP/IP Checksum Support

The 3C90xB NICs provide automatic TCP/IP checksum insertion and verification. A packet in the transmit FIFO may be scanned to determine if it is an IP version 4 packet, and if so, whether it contains a TCP or UDP segment header. Checksums may then be calculated and inserted into the headers.

The host driver requests TCP/IP checksumming for a packet by setting one or more checksum enable bits in the packet's Frame Start Header DPD entry.

TCP/IP checksumming may delay transmission. A packet must be completely downloaded to calculate checksums, and transmission cannot commence until the checksums are calculated and inserted.

On reception, every packet is scanned for the presence of IP, TCP, and UDP headers. For any that are found, checksums are calculated and compared with those contained in the packet. Any mismatches are flagged in the checksum error bits in the UPD's Up Pkt Status entry.

The following restrictions apply to TCP/IP checksum support:

- IP version 4 only. Packets that contain other IP versions are ignored by the checksumming hardware.
- IP forms: (EtherType = 0800), 802.2, or SNAP.
- No support for reception of fragmented IP datagrams. Fragmented datagrams have their IP header checksums verified, but the TCP or UDP checksums are ignored. The fourth word in an IP header contains a 13-bit Fragment Offset and a More Fragments bit. If the Fragment Offset is nonzero or the More Fragments bit is set, the packet is considered a fragmented datagram.
- Support for IEEE 802.1Q VLANs. When the checksumming logic encounters a packet in which the thirteenth and fourteenth bytes match the VlanEtherType register, it recognizes the packet as IEEE 802.1Q and skips the thirteenth through sixteenth bytes in its checksum calculation.
- Support for VLT packets. When the vltEnable bit in the MacControl register is set, the checksumming logic skips the first through fourth bytes in every packet during its checksum calculation.

# 4 CONFIGURATION

This chapter discusses the NIC configuration mechanism and defines the registers associated with configuration. Configuration has two components: NIC configuration and PCI configuration.

## System Reset

System reset is the assertion of the hardware reset signal on the PCI bus, which causes a complete reset of the NIC, including forcing flip-flops to known values, losing any NIC configuration that had been set, and loading the default configuration from the EEPROM.

There are two sources of system reset:

- A hardware reset, caused by asserting the PCI Reset signal after power-up, which brings the NIC into a known state.
- A software-controlled reset, using the GlobalReset command in the Command register. A GlobalReset command bit mask parameter allows selective reset of various parts of the NIC.

For details on GlobalReset and other reset commands, see "Reset Commands" in Chapter 10.

## Serial EEPROM

The serial EEPROM is used for nonvolatile storage of such information as the DeviceId, node address, manufacturing data, default configuration settings, and software information.

Some of the EEPROM data (such as the DeviceId and configuration defaults) is automatically read into the NIC logic after system reset, whereas other data (such as the node address and software information) is meant to be read by driver software.

Shortly after system reset, the NIC ASIC reads certain locations from the EEPROM and places the data into the host-accessible registers shown in Table 11.

**Table 11** EEPROM Data Locations

| EEPROM Location | Register | Register Type |
|---|---|---|
| 3 | DeviceId | PCI configuration |
| 12 | InternalConfig Word 0 | I/O |
| 13 | InternalConfig Word 1 | I/O |
| 17 | SubsystemVendorId | PCI configuration |
| 18 | SubsystemId | PCI configuration |
| 19 | MediaOptions | I/O |

## NIC Configuration

PCI NICs use a slot-specific block of configuration registers to perform NIC configuration. The configuration registers are accessed with two types of PCI configuration cycles:

- Type 0 cycles are used to configure devices on the local PCI bus.
- Type 1 cycles are used to pass a configuration request to a PCI bus at a different hierarchical level.

PCI configuration cycles are directed at one out of eight possible PCI logical functions within a single physical PCI device.

*The 3C90x NICs respond only to Type 0 configuration cycles, directed at function 0. The NICs ignore Type 1 cycles and Type 0 cycles directed at functions other than 0. The 3C90xB NICs respond to both types of configuration cycles.*

Each PCI device decodes 256 bytes worth of configuration registers. Of these, the first 64 bytes are predefined by the PCI specification. The remaining registers may be used as needed for PCI device-specific configuration registers.

In PCI configuration cycles, the host system provides a slot-specific decode signal (IDSEL) that informs the NIC that a configuration cycle is in progress. The NIC responds by asserting DEVSEL# and decoding the specific configuration register from the address bus and the byte enable signals. See the *PCI BIOS Specification* (available from the BIOS vendor) for information on generating configuration cycles from driver software.

Configuration consists of allocating system resources to the NIC and setting NIC-specific options. This is done by writing values into special PCI configuration registers and into I/O registers. The location of this configuration space in the host processor's address map is system-dependent.

- For 3C90x NICs, PCI configuration is performed by a BIOS routine supplied with the computer system.
- For 3C90xB NICs, PCI configuration is performed by a POST routine supplied with the computer system.

NIC-specific configuration is the driver's responsibility. The registers that are set during PCI configuration are summarized in Table 12.

**Table 12** PCI Registers Set During Configuration

| | Configured by | | |
|---|---|---|---|
| **Register** | **BIOS** | **POST** | **Description** |
| PciCommand | | X | Enables NIC operation through response to and generation of PCI bus cycles, and enables parity error generation. |
| IoBaseAddress | X | | Sets the I/O base address for the NIC registers. |
| MemBaseAddress | X | | Sets the memory base address for the NIC registers. |
| BiosRomControl | X | | Sets the base address and size for an optional installed expansion ROM. |
| CacheLineSize | X | | Indicates the system's cache line size. The NIC uses this value for optimizing bus master data transfers. |
| LatencyTimer | X | | Programs a NIC timer that controls how long the NIC can hold the bus as a bus master. |
| InterruptLine | X | | Maps the NIC's interrupt request to a specific interrupt line (level) on the system board. |
| InternalConfig | | X | Selects the media port (transceiver) and local RAM parameters. The InternalConfig register is mapped into window 3 of the I/O register space. |

## Forced Configuration

The 3C90x and 3C90xB NICs include a forced-configuration mode that enables them to be accessed across the PCI bus without first needing to perform PCI configuration or to load data from the EEPROM. Forced configuration mode is intended for NIC testing only.

Forced configuration mode is useful for embedded applications in which it is desired to operate the chip without an EEPROM.

In forced configuration mode, the NIC is forced to the following configuration settings:

- I/O base address: 200h
- I/O target cycles: enabled
- Memory target cycles: disabled
- Bus master cycles: enabled
- BIOS ROM cycles: disabled

The NIC is placed into forced configuration mode as a result of an external pin strapping.

## Support for Signaling Standards

The NIC ASICs support the following signaling standards:

- The two physical signaling schemes defined in the IEEE 802.3u Fast Ethernet specification
- The three legacy 10 Mbps IEEE 802.3 signaling standards

Three basic media types are supported:

- 10 Mbps
- 100BASE-X (FX and TX)
- Media-independent Interface (MII)

A signal multiplexer selects which media type is active by connecting it to the Ethernet MAC, which is the common point for all of the media types.

The signal multiplexer is controlled by a combination of the xcvrSelect field in the InternalConfig register and the IEEE 802.3u auto-negotiation function.

Media port architecture details for the NIC ASICs appear in Figure 21 through Figure 24.

**Figure 21** 3C90x NICs (40-0336-00x ASIC) Media Port Architecture

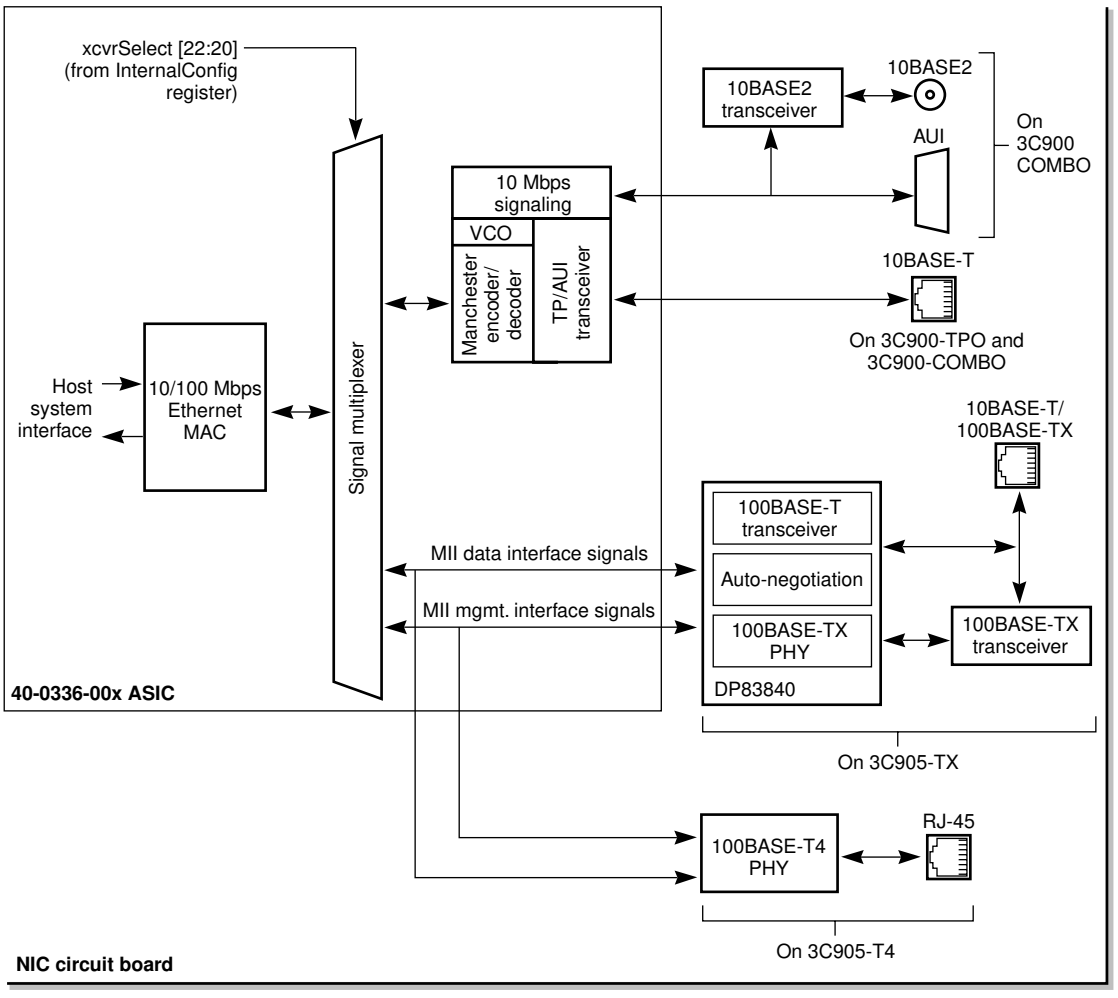


**Figure 22** 3C900B NICs (40-0456-004 ASIC) Media Port Architecture

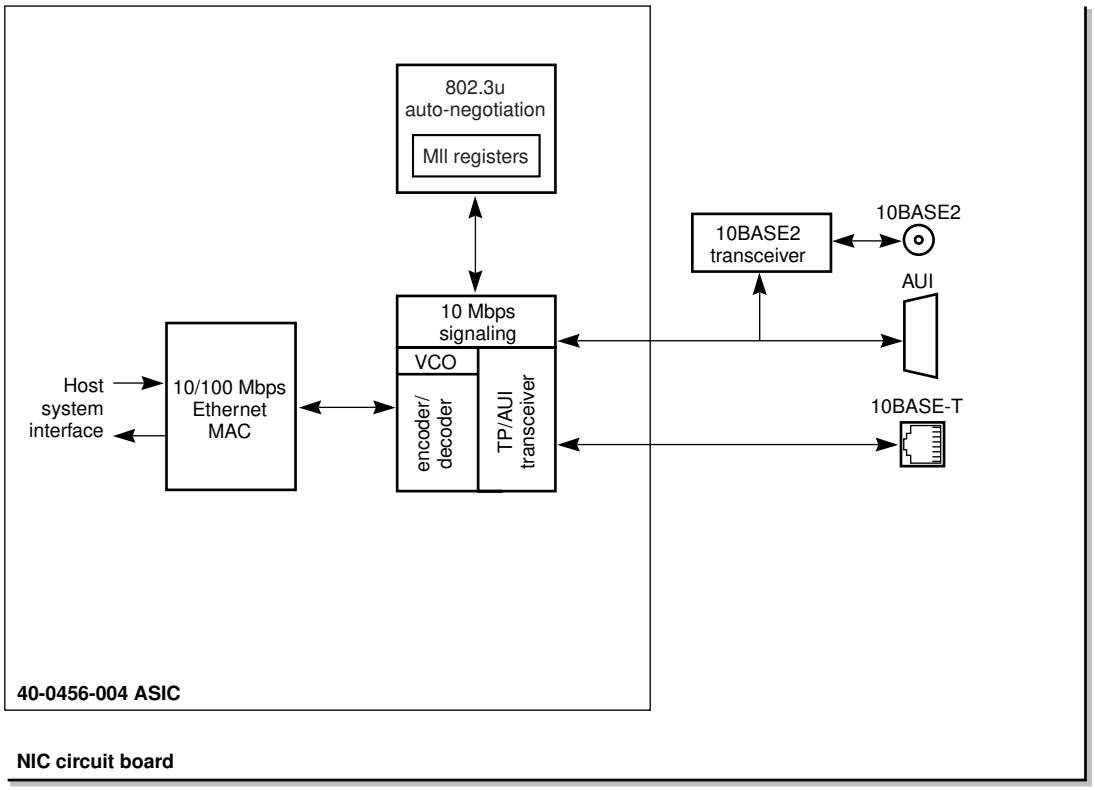


**Figure 23** 3C90xB NICs (40-0502-00x ASIC) Media Port Architecture

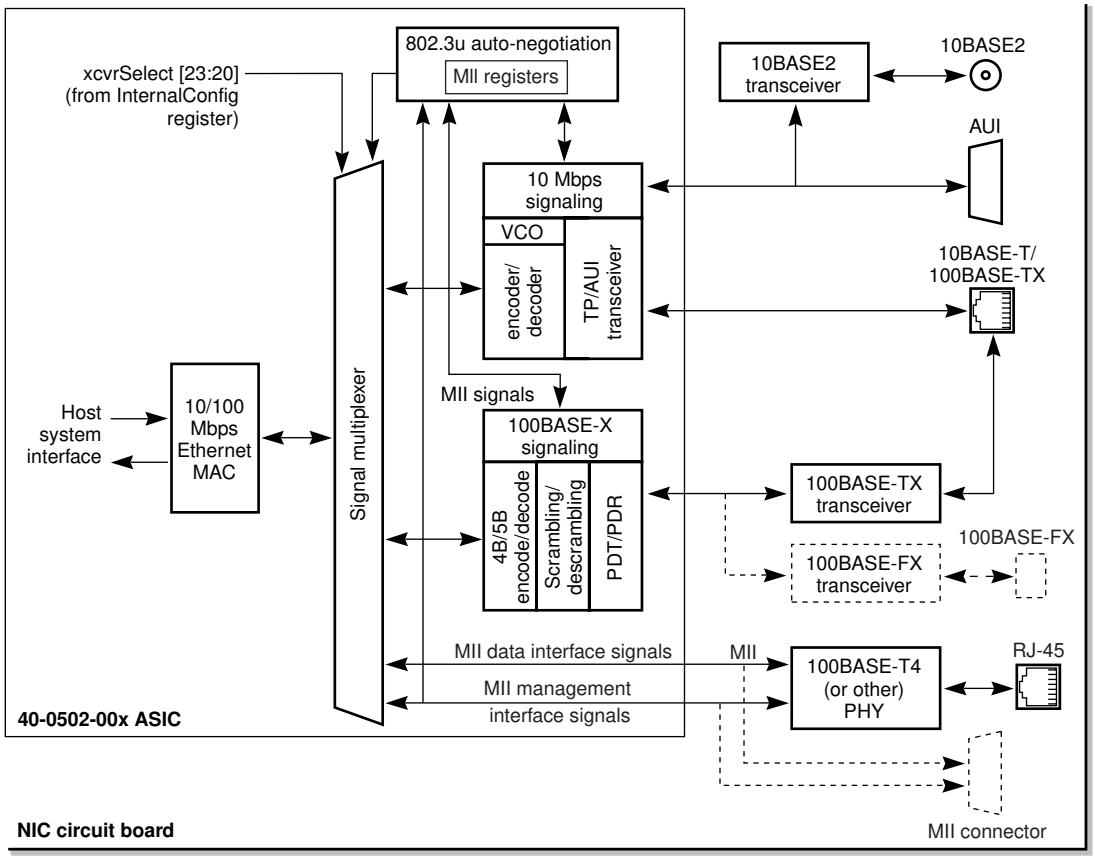
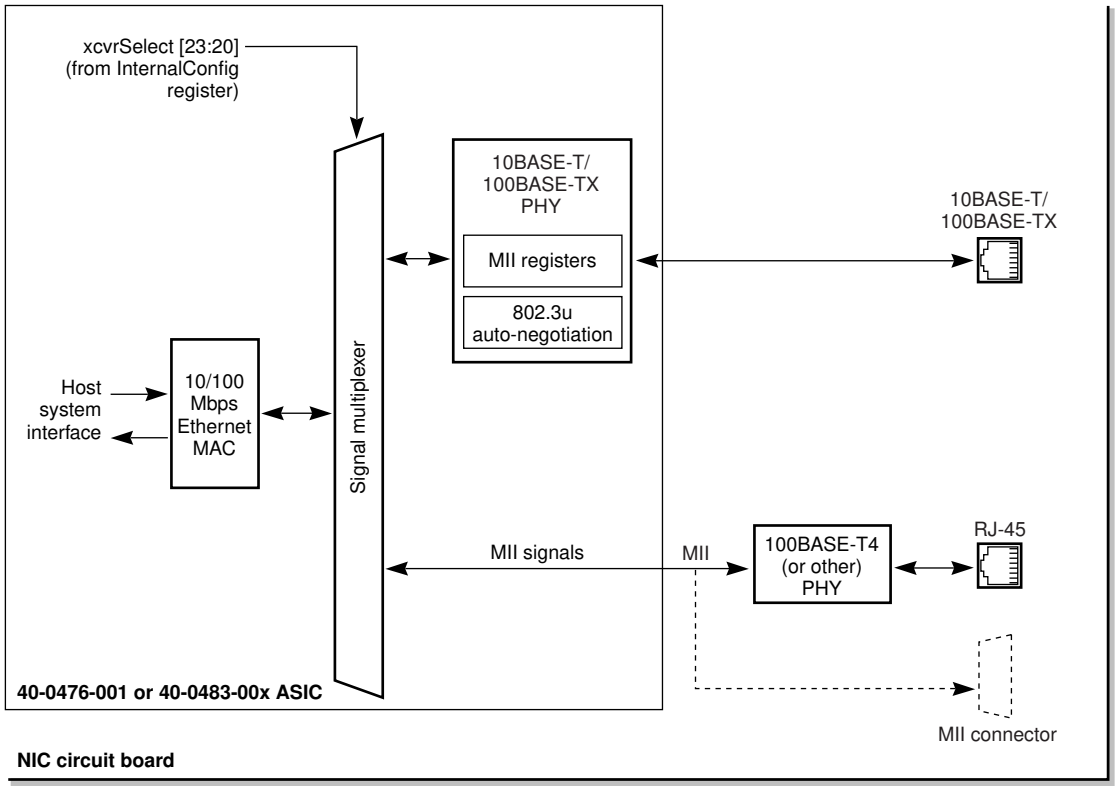



**Figure 24** 3C905B-TX NICs (40-0476-001 or 40-0483-00x ASIC) Media Port Architecture



### 10 Mbps Signaling

10 Mbps signaling supports the following types of connections:

- 10BASE2—this signaling requires a coax transceiver external to the ASIC.
- AUI—the ASIC connects directly to the AUI D-shell connector.
- 10BASE-T—magnetics connect the ASIC to the RJ-45 connector. With additional circuitry, 10BASE-T and 100BASE-TX can share the same RJ-45 connector.

The NIC includes the Manchester encoder/decoder, clock recovery VCO, and the line drivers/receivers.

### 100BASE-X Signaling

The 100BASE-X standard combines the IEEE 802.3 CSMA/CD Media Access Control (MAC) specification and FDDI PMD specifications. There are two types of 100BASE-X:

- 100BASE-TX defines signaling over two pairs of Category 5 twisted-pair wiring, as defined in the *ANSI FDDI TP-PMD Specification*.
- 100BASE-FX defines signaling over FDDI-standard fiber-optic cabling, as defined in the *ANSI FDDI PMD Specification*.

The NIC ASICs include the 4B/5B encoding, decoding, scrambling, descrambling, and clock generation/clock recovery functions required for 100BASE-X. Only an external transceiver is required for a complete 100BASE-X solution. From the perspective of the NIC ASICs, the only difference between 100BASE-TX and 100BASE-FX is that scrambling and descrambling are disabled for 100BASE-FX operation.

100BASE-TX is typically implemented to share an RJ-45 connector with 10BASE-T. This requires some switching circuitry external to the NIC ASICs, and the driver must select the media speed.

### Media-Independent Interface/100BASE-T4

MII provides a general-purpose interface between an IEEE 802.3u MAC and various physical layer devices. MII has two components:

- A data interface that provides separate 4-bit-wide paths for receive data and transmit data. The MII data interface is connected to the Ethernet MAC by programming the code for MII in the xcvrSelect field of the InternalConfig register.
- A management interface that is a bidirectional serial link that provides access to a physical layer device's internal registers. Driver software controls the management interface through bits in the NetworkDiagnostic register. For information about programming management interface accesses, see Appendix B.

The typical application for the MII is to provide 100BASE-T4 support. 100BASE-T4 refers to a system including the IEEE 802.3u CSMA/CD MAC and signaling over four pairs of Category 3 (or better) twisted-pair wiring. For information on 100BASE-T4 PHY registers, refer to the following World Wide Web site:

**`http://www.broadcom.com`**

Because the MII is independent of the signaling method, it is possible to use it to support any type of 10 Mbps or 100 Mbps signaling, depending on the availability of MII-compliant PHY devices.

## Auto-Negotiation

IEEE 802.3u auto-negotiation provides automatic negotiation of signaling rate and duplex mode between the two ends of a twisted-pair link segment. Typically, the two ends of a link segment are an end station (NIC) and a hub or switch.

IEEE auto-negotiation is specified to negotiate the following signaling technologies: 10BASE-T, 100BASE-TX, and 100BASE-T4.

The auto-negotiation function interacts with the 10 Mbps and 100BASE-X functions to negotiate a common operating mode with its link partner. If a common mode is found and a link is established, auto-negotiation directs the signal multiplexer to connect the appropriate signaling function to the MAC.

*On 3C90x NICs, the auto-negotiation function is implemented within the National Semiconductor DP83840 PHY chip. For details on that chip, refer to the DP83840 specification from National Semiconductor.*

On the 3C90xB NICs, an integrated IEEE 802.3u auto-negotiation function handles auto-negotiation for 10BASE-T and 100BASE-TX media types. 100BASE-T4 is not implemented within the NICs.

*There are three different versions of the ASIC on the 3C905B-TX NIC. These three ASICs function identically except for the PHY portion, which is proprietary to each ASIC. As a result, the MII register layouts differ on the three ASICs. See Chapter 11 for information on the MII register layouts of each ASIC.*

For more information on the ASICs, see "ASICs" in Chapter 2.

For more details on auto-negotiation, see Chapter 11.

## BIOS ROM

The NICs support an optional BIOS ROM through a socket. The following Atmel PEROM flash devices are supported:

- AT29C512 (64K × 8)
- AT29C010 (128K × 8)

The BIOS ROM is configured through the BiosRomControl PCI configuration register. This register causes the ROM to be mapped into the memory space of the host system, allowing the ROM contents to be scanned, copied to system RAM, and executed at initialization time.

The BIOS ROM in memory space is read-accessible using byte, word, or double-word cycles. All write accesses to the BIOS ROM in memory space require double-word writes to the NIC.

The BIOS ROM is also byte-read and byte-write accessible to the host CPU through the BiosRomData and BiosRomAddr I/O registers. This allows a diagnostic program to read or modify the ROM contents without having to write to configuration registers.

## InternalConfig

| | |
|---|---|
| Synopsis | Allows the setting of a NIC-specific configuration. |
| Type | Read/write |
| Size | 32 bits |
| Window | 3 |
| Offset | 0 |

The InternalConfig register provides a way to set NIC-specific, non-host-related configuration settings. The contents of InternalConfig are read from EEPROM at reset.

Two words supply the value for InternalConfig:

- InternalConfig word 0 corresponds to the low-order 16 bits of the register.
- InternalConfig word 1 corresponds to the high-order 16 bits of the register.

The least-significant word of InternalConfig contains hardware configuration information that should generally not be changed by software.

The most-significant word contains information that may be changed by installation software to tune the NIC to the system configuration. It also has a field to select the media port.

InternalConfig is set to 00000010h at reset but is normally loaded with a value from EEPROM shortly after reset.

*The InternalConfig register differs for the 3C90x and 3C90xB NICs. The register formats and bit descriptions for each type of NIC are described below.*

**3C90x NICs InternalConfig Register Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | | | | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 | | | | | 0 | 0 | | | | |

**3C90x NICs InternalConfig Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [2:0] | ramSize | This field specifies the size of the packet buffer SRAM installed on the NIC (or inside the ASIC), as defined below:<br>■ 0 = 8 KB RAM<br>■ 2 = 32 KB RAM<br>■ 3 = 64 KB RAM<br>■ 4 = 128 KB RAM |
| [3] | ramWidth | This read-only field specifies the width of the packet buffer RAM, as defined below:<br>■ 0 = Byte-wide<br>■ 1 = Word-wide |

**3C90x NICs InternalConfig Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [7:6] | romSize | Specifies the size of the BIOS ROM installed on the NIC, as defined below:<br>■ 0 = 8 KB<br>■ 1 = 16 KB<br>■ 2 = 32 KB<br>■ 3 = 64 KB RAM |
| [8] | disableBadSsdDetect | This bit, when set, disables checking for a proper JK symbol in the first byte of the 100BASE-TX/FX start of stream delimiter (SSD). When this bit is clear, the NIC checks for a proper JK in this byte, and updates the BadSSD counter accordingly (if statistics are enabled).<br>This bit only has an effect in 100BASE-TX/FX modes. |
| [9] | ramLocation | This bit, when cleared, specifies that external RAM should be used. When set, it specifies that internal RAM should be used. |
| [17:16] | ramPartition | This field specifies how the packet buffer RAM is divided between the receive and transmit FIFOs, as defined below. See also Table 13.<br>■ 0 = 5:3<br>■ 1 = 3:1<br>■ 2 = 1:1<br>■ 3 = 3:5<br>The valid values of ramPartition for various ramSize and ramWidth combinations are shown in Table 13. |
| [22:20] | xcvrSelect | This read/write field indicates the selected transceiver type. The only legal values for xcvrSelect are those that have a corresponding bit set in the ResetOptions register.<br>After changing the value of xcvrSelect, drivers should issue both the RxReset and TxReset commands.<br>The transceiver types are defined below: |
| | **xcvrSelect Value** | **Transceiver Selected** |
| | 000 | 10BASE-T |
| | 001 | 10 Mbps AUI: Configure the NIC for AUI operation. The specific type of MAU (transceiver) operating over the AUI is unspecified. |
| | 010 | Reserved. |
| | 011 | 10BASE2: Configure the NIC for 10BASE2 operation. Note that when 10BASE2 is chosen, the EnableDcConverter command also must be issued before network operation can begin. |
| | 100 | Reserved. |
| | 101 | 100BASE-FX: Configure the NIC for 100BASE-FX operation. |

**3C90x NICs InternalConfig Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| | 110 | MII: Configure the NIC for MII operation. The specific type of PHY device (transceiver) operating over the MII is unspecified. The MII PHY device may or may not have its own auto-negotiation entity. If it has its own auto-negotiation entity, it is functionally distinct from the 3C90xB NICs' integrated auto-negotiation function.<br>MII is a general-purpose port that can be used to support 100BASE-T4 signaling. |
| | 111 | Reserved. |
| [24] | autoSelect | When set, indicates that the driver should ignore the value set in xcvrSelect, and instead auto-select the media port at load time. If autoSelect is clear, the xcvrSelect value is used as is, and the driver configures the NIC accordingly.<br>Although this bit is read/write, drivers should treat it as read-only. |

(3 of 3)

**Table 13** 3C90x NIC ramPartition Values

| ramSize | ramWidth | |
|---|---|---|
| | **Byte** | **Word** |
| 8 KB | 1:1, 3:1, 5:3, 3:5 | 1:1, 3:1, 5:3, 3:5 |
| 32 KB | 1:1 | — |
| 64 KB | — | 1:1, 3:1 |
| 128 KB | — | 1:1 |

**3C90xB NICs InternalConfig Register Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | 0 | 0 | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | | | | 0 | 0 | 0 | 0 | 0 | 0 |

**3C90xB NICs InternalConfig Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [7:6] | romSize | Specifies the size of the BIOS ROM installed on the NIC, as defined below:<br>■ 0 = 64 KB<br>■ 1 = 128 KB<br>■ 2, 3 = Reserved |
| [8] | disableBadSsdDetect | This bit, when set, disables checking for a proper JK symbol in the first byte of the 100BASE-TX/FX start of stream delimiter (SSD).<br>When this bit is clear, the NIC checks for a proper JK in this byte, and updates the BadSSD counter accordingly (if statistics are enabled).<br>This bit only has an effect in 100BASE-TX/FX modes. |
| [14] | enableTxLarge | This read/write bit, when set, enables transmission of packets that are larger than the transmit FIFO.<br>Because the 3C90xB NICs' FIFO size is 2 KB, this bit can be left clear (the reset default). |

(1 of 3)

**3C90xB NICs InternalConfig Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [15] | enableRxLarge | This read/write bit, when set, enables reception of packets that are larger than the receive FIFO.<br><br>Because the 3C90xB NICs' FIFO size is 2 KB, this bit can be left clear (the reset default). |
| [23:20] | xcvrSelect | This read/write field indicates the selected transceiver type. On a given NIC product, the only legal values for xcvrSelect will be those that have a corresponding bit set in the MediaOptions register.<br><br>After changing the value of xcvrSelect, drivers must issue both an RxReset and a TxReset command.<br><br>The transceiver types are defined below: |
| | **xcvrSelect Value** | **Transceiver Selected** |
| | 0000 | 10BASE-T: This setting is used for backward compatibility and diagnostic purposes only. Drivers wishing to operate over 10BASE-T should set these bits to the auto-negotiation setting and configure the auto-negotiation registers for the appropriate mode. |
| | 0001 | 10 Mbps AUI: Configure the NIC for AUI operation. The specific type of MAU (transceiver) operating over the AUI is unspecified. |
| | 0010 | Reserved. |
| | 0011 | 10BASE2: Configure the NIC for 10BASE2 operation. Note that when 10BASE2 is chosen, the EnableDcConverter command also must be issued before network operation can begin. |
| | 0100 | 100BASE-TX: This setting is used for backward compatibility and diagnostic purposes only. Drivers that need to operate over 100BASE-TX should set these bits to the auto-negotiation setting and configure the auto-negotiation registers for the appropriate mode. |
| | 0101 | 100BASE-FX: Configure the NIC for 100BASE-FX operation. |
| | 0110 | MII: Configure the NIC for MII operation. The specific type of PHY device (transceiver) operating over the MII is unspecified. The MII PHY device may or may not have its own auto-negotiation entity. If it has its own auto-negotiation entity, it is functionally distinct from the integrated auto-negotiation function of the NIC.<br><br>The MII is a general-purpose port that can be used to allow the NIC to signal over a variety of media types. The MII is used to support (among other things) 100BASE-T4 signaling. |
| | 0111 | Reserved. |

**3C90xB NICs InternalConfig Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| | 1000 | Auto-negotiation: The auto-negotiation function automatically determines the data rate and duplex mode.<br><br>After auto-negotiation, the chip is automatically configured to operate at the negotiated data rate. However, if a full-duplex link was negotiated, the driver must configure the duplex mode by setting the fullDuplexEnable bit in the MacControl register.<br><br>The on-chip auto-negotiation only handles 10BASE-T and 100BASE-TX connections. In an AutoSelect sequence, if auto-negotiation is unable to establish a link, then the driver needs to try the other supported media types. |
| | 1001 | MII external MAC mode. |
| | 1010–1111 | Reserved. |
| [24] | autoSelect | When set, this bit indicates that the driver should ignore the value set in xcvrSelect, and instead auto-select the media port at load time. If autoSelect is clear, the xcvrSelect value is used as is, and the driver configures the NIC accordingly.<br><br>Although this bit is read/write, it should be treated as read-only by drivers. |
| [25] | disableBiosROM | This bit, when set, disables accesses to the on-NIC BIOS ROM. This bit is included to allow bypassing the BIOS ROM without having to physically remove it from the board. When this bit is set, the NIC responds to any read in its configured BIOS ROM space by returning 00000000h, and it ignores writes to the BIOS ROM. |

(3 of 3)

## NIC Initialization

After the system has performed basic configuration of the NIC, software must initialize the NIC, which means selecting the media port and setting various NIC registers to the desired initial values.

### Selecting the Media Port

The media port (transceiver) can be selected two ways:

- Through the EEPROM (by using the xcvrSelect field in the InternalConfig register)
- Through automatic selection (by using AutoSelect)

#### Selection Through EEPROM

Because the value of the InternalConfig register is stored in the serial EEPROM and loaded into the ASIC after reset, it is possible to write an xcvrSelect value into EEPROM and thereafter have the NIC automatically use the stored value when it is powered up.

The xcvrSelect field settings enable specific ports. After the value of xcvrSelect is changed, drivers must always issue RxReset and TxReset commands.

#### AutoSelect

Alternatively, an AutoSelect mechanism causes the driver to ignore the EEPROM value for xcvrSelect and attempt to set the media port based on which port is currently active.

When the autoSelect bit in the InternalConfig register is set, the driver selects each port available on the NIC in turn (see "MediaOptions", the next section). The driver attempts to find a port that is connected to the network. Different kinds of ports require different techniques to determine an active link. If the driver fails to find a connected port, it restores the original value in the xcvrSelect field.

For more details on autoSelect, see "AutoSelect Sequence""in this chapter.

**MediaOptions**

The MediaOptions register provides a way for driver or configuration software to determine the media ports installed on a 3C90xB NIC.

The value for MediaOptions is stored in word 19 of the EEPROM and is read into the ASIC after a reset. During an AutoSelect sequence, a driver checks MediaOptions to determine which ports it should try.

*The 3C90x NICs use the ResetOptions register to determine the media ports installed on a NIC.*

**AutoSelect Sequence**

The following paragraphs describe, in general terms, the techniques for determining the active media port. For detailed AutoSelect pseudo code, see Appendix A.

### Auto-Negotiation

The IEEE 802.3u auto-negotiation logic attempts to negotiate a 10BASE-T or 100BASE-TX link with the hub or switch. Shortly after reset or power-up, the auto-negotiation logic starts the auto-negotiation process. This consists of advertising the NIC's capabilities using encoded fast link pulses (FLPs) and listening for indications of the link partner's capabilities.

Auto-negotiation is designed to allow the NIC to establish a valid link with the following hubs or switches:

- Those that implement auto-negotiation
- Those that are pre–auto-negotiation 10BASE-T
- Those that are pre–auto-negotiation 100BASE-TX

Typically, by the time driver software is loaded, auto-negotiation is finished and the driver is able to determine the negotiated link speed and mode. However, because the driver is unable to distinguish between an auto-negotiation incomplete state and an auto-negotiation failure state, it must implement a timeout loop to wait a reasonable time period before deciding that auto-negotiation has failed.

A driver typically executes the following steps in its auto-negotiation sequence.

**1** Check that auto-negotiation is complete.

For example, the 3C90xB NIC checks that bit 5 (autoNegComplete) in the MII register 01 (AutoNegStatus) is set, indicating that auto-negotiation is complete. If not complete, execute a timeout loop of some reasonable length, and check again.

2 Compare the advertised capabilities of the NIC against the capabilities received from the link partner.

   For 3C90xB NICs, this involves doing a bitwise AND on bits [5:8] of MII registers 04 and 05 (AutoNegAdvert and AutoNegAbility).

   The highest common capability is the negotiated mode. If no common capability is found, no link has been established.

3 If a full-duplex mode has been negotiated, set the fullDuplexEnable bit in the MacControl register, if desired.

#### MII/100BASE-T4

Any number of different PHY devices can be connected to the MII port. Each device either supports auto-negotiation or has its own proprietary link detection sequence. In general, driver software must communicate with the PHY device registers across the MII management interface to determine the link status of the PHY.

The typical application for MII is to support 100BASE-T4. Current 100BASE-T4 devices have a proprietary scheme for determining and selecting the active port. Future 100BASE-T4 PHY devices may support IEEE 802.3u auto-negotiation. Refer to the PHY device specifications for more details.

#### 100BASE-FX

100BASE-FX is not covered by auto-negotiation and cannot be looped back, so the link is tested using a combination of checking the linkDetect bit in the MediaStatus register and listening for receive frames.

To check for a 100BASE-FX link:

1 Set the xcvrSelect bit in the InternalConfig register to 100BASE-FX.

2 Issue RxReset and TxReset commands.

3 Set the linkBeatEnable bit in the MediaStatus register.

4 Set the receiveAllFrames bit in the RxFilter register (promiscuous mode) and issue RxEnable and TxEnable commands.

5 Transmit a self-directed packet (some hubs require this to unpartition the link so that they can receive packets).

6 Enter a loop that looks for any receive packets, reads the linkDetect bit in MediaStatus, and checks the carrierSense bit. The number of times the carrierSense bit is seen active is accumulated. If at any time in the loop a packet is received without error, the link is considered good. If error packets are received, they are discarded and the loop continues. If at any time the linkDetect bit is off, the link is considered bad.

   If the loop completes a large number of iterations without ever seeing linkDetect off or receiving a good frame, then the carrierSense statistic is used to guess the link status. If carrierSense was detected "on" for more than 25% of the loop iterations, then on a good link this should have resulted in a good receive frame, so the link is considered bad. If carrierSense was seen active less than 25% of the time, then carrierSense is considered a false reading and the link is declared good.

#### AUI

To check for an active AUI link:

1. Set the xcvrSelect bit in the InternalConfig register to AUI.
2. Issue RxReset and TxReset commands.
3. Configure the NIC for external loopback by setting the fullDuplexEnable bit in the MacControl register, enabling the transmitter and receiver, and setting the StationAddress register and receive filter.
4. Transmit a self-directed test packet.

   If the same packet is received, it indicates a properly terminated shared-medium network (typically 10BASE5) is connected. Note that other media types may be connected through the AUI port, which may require some other method to test for a valid network.

#### 10BASE2

To check for an active 10BASE2 link:

1. Set the xcvrSelect bit in the InternalConfig register appropriately.
2. Issue RxReset and TxReset commands.
3. Issue the EnableDcConverter command to turn on the DC/DC converter (to power the 10BASE2 transceiver).
4. Because there is no link test on 10BASE2, the link is tested by attempting to send a loopback packet. Configure the NIC for external loopback by setting the fullDuplexEnable bit in the MacControl register, enabling the transmitter and receiver, and setting the StationAddress register and receive filter. Then transmit a self-directed packet. If the same packet is received, it indicates that a proper, terminated 10BASE2 network is connected.

   If no valid network is found, be sure to issue the DisableDcConverter command before trying any other media ports.

#### Manual Testing of 10BASE-T and 100BASE-TX

The approach outlined for 100BASE-FX can also be applied to manual testing of the 10BASE-T and 100BASE-TX ports, if it is desired to bypass the auto-negotiation process. Use the 10BASE-T and 100BASE-TX choices in the xcvrSelect field, instead of auto-negotiation.

## Setting the Receive Filter

The RxFilter register determines which types of packets can be received by the NIC. RxFilter is set using the SetRxFilter command.

### Station Address

The driver is expected to program the NIC's network address into the StationAddress register. The NIC's network address can be obtained from the appropriate data locations within the EEPROM. The driver is, of course, free to program any arbitrary value into StationAddress.

Once StationAddress is programmed, the NIC can be configured to receive packets whose destination addresses match that address by setting the receiveIndividual bit in RxFilter.

**Broadcast Packets** Setting the receiveBroadcast bit in the RxFilter register causes the NIC to receive all broadcast packets.

**Multicast Packets** Setting the receiveMulticast bit in the RxFilter register causes the NIC to receive all multicast packets.

**Multicast Address Hash Filter** The 3C90xB NICs contain a hash filter for selective reception of multicast packets. The hash filter is an array of bits:

- 64 on 3C900B NICs and 3C90xB NICs with the 40-0502-00x ASIC
- 256 on 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC

The NIC applies a cyclic redundancy check (CRC) to the destination address of incoming packets that have the multicast bit set. The low-order eight bits (six bits on 3C90xB NICs) of the CRC are used as an index into the hash filter. If the hash filter bit addressed by the index is set, the packet is accepted by the NIC and is passed to the driver. If the hash filter bit is cleared, the NIC discards the packet.

Setting the receiveMulticastHash bit in the RxFilter register enables the filtering mechanism. The hash filter is programmed using the SetHashFilterBit command.

**Promiscuous Mode** Setting the receiveAllFrames bit in the RxFilter register causes the NIC to receive all packets in promiscuous mode.

## Capabilities Word

This word is a 16-bit location in the EEPROM that specifies the capabilities of the NIC.

See the "Data Format" section in Chapter 5 for more details.

## MacControl

The MacControl register is used to configure MAC-specific parameters, including full-duplex mode, flow control enabling, and extended deference options.

## Setting the Duplex Mode

The NIC can operate in either half-duplex or full-duplex mode.

- In half-duplex mode, the NIC cannot transmit and receive simultaneously. Before it can transmit, it must wait for the network link to go quiet, and a collision may occur once transmission has begun.
- In full-duplex mode, the NIC can transmit and receive simultaneously, and collisions do not exist.

**3C90x NICs** On 3C90x NICs, the duplex mode is set entirely by the fullDuplexEnable bit in the MacControl register. This bit controls the duplex mode even when the NIC uses auto-negotiation to establish the link. If the auto-negotiation process results in a full-duplex link (as indicated by the baseTFullDuplex and baseTxFullDuplex bits in the AutoNegAdvert and AutoNegAbility registers), the driver must sense this and set fullDuplexEnable, if desired.

**3C90xB NICs**

On 3C90xB NICs, the duplex mode is set by a combination of fullDuplexEnable in the MacControl register and the auto-negotiation bits in the MII registers. If auto-negotiation is not enabled (xcvrSelect in InternalConfig is not set to 1000b), fullDuplexEnable fully controls the duplex mode. In this case, the criteria for setting fullDuplexEnable are driver- and environment-dependent.

If auto-negotiation is enabled (xcvrSelect is set to 1000b), then the negotiated duplex mode is indicated by bits 6 and 8 in MII registers 04 and 05 (AutoNegAdvert and AutoNegAbility). If these bits indicate that a full-duplex link has been negotiated, the driver must sense this and set fullDuplexEnable.

## PCI Configuration Registers

Table 14 and Table 15 summarize the PCI configuration registers implemented by the NICs.

Shaded spaces and all locations within the 256-byte configuration space that are not shown in the tables are reserved or not implemented and return zero when read.

**Table 14** Summary of 3C90x NICs PCI Configuration Registers

| Byte 3 | Byte 2 | Byte 1 | Byte 0 | Offset |
|---|---|---|---|---|
| | | EepromData | | 4c |
| | | ResetOptions | | 48 |
| | | | | 44 |
| InternalConfig | | | | 40 |
| MaxLat | MinGnt | InterruptPin | InterruptLine | 3c |
| | | | | 38 |
| | | | | 34 |
| BiosRomControl | | | | 30 |
| | | | | 2c |
| | | | | 28 |
| | | | | 24 |
| | | | | 20 |
| | | | | 1c |
| | | | | 18 |
| | | | | 14 |
| IoBaseAddress | | | | 10 |
| | HeaderType | LatencyTimer | | 0c |
| ClassCode | | | | 08 |
| PciStatus | | PciCommand | | 04 |
| DeviceId | | VendorId | | 00 |

**Table 15** Summary of 3C90xB NICs PCI Configuration Registers

| Byte 3 | Byte 2 | Byte 1 | Byte 0 | Offset |
|---|---|---|---|---|
| | | PowerMgmtCtrl | | e0 |
| PowerMgmtCap | | NextPtr | CapID | dc |
| | | | | 60 – d8 |
| | | | MedTestPattern | 5c |
| | | | MedTestOutput | 58 |
| | | | MedTestModeHi | 54 |
| | | | MedTestModeLo | 50 |
| | | | | 4c |
| | | | | 48 |
| | | | | 44 |
| | | | | 40 |
| MaxLat | MinGnt | InterruptPin | InterruptLine | 3c |
| | | | | 38 |
| | | | CapPtr | 34 |
| BiosRomControl | | | | 30 |
| SubsystemId | | SubsystemVendorId | | 2c |
| | | | | 28 |
| | | | | 24 |
| | | | | 20 |
| | | | | 1c |
| | | | | 18 |
| MemBaseAddress | | | | 14 |
| IoBaseAddress | | | | 10 |
| | HeaderType | LatencyTimer | CacheLineSize | 0c |
| ClassCode | | | RevisionId | 08 |
| PciStatus | | PciCommand | | 04 |
| DeviceId | | VendorId | | 00 |

The following sections describe the PCI configuration registers.

**VendorId** This read-only register contains the unique 16-bit manufacturer's ID as allocated by the PCI SIG. 3Com's manufacturer ID is 10B7h.

**DeviceId** This read-only register contains the 3Com-allocated 16-bit device ID for the NIC. This value is read from EEPROM location 03 after reset.

**PciCommand** This read/write register provides control over the NIC's ability to generate and respond to PCI cycles. When a zero is written to this register, the NIC is logically disconnected from the PCI bus, except for configuration cycles.

**PciCommand Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | | 0 | | 0 | | | |

**PciCommand Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | ioSpace | Setting this bit allows the NIC to respond to I/O space accesses (if the NIC is in the D0 power state). |
| [1] | memorySpace | Setting this bit (along with the addressDecodeEnable bit in the BiosRomControl register) allows the NIC to decode accesses to its BIOS ROM, if one is installed, and if the NIC is in the D0 power state. |
| [2] | busMaster | Setting this bit allows NICs with bus master capability to initiate bus master cycles (if the NIC is in the D0 power state). |
| [4] | MWIEnable | Memory Write and Invalidate Enable. Setting this bit allows the NIC to generate the MWI command. |
| [6] | parityErrorResponse | This bit controls how the NIC responds to parity errors. Setting this bit causes the NIC to take its normal action upon detecting a parity error. Clearing this bit causes the NIC to ignore parity errors. This bit is cleared upon system reset. |
| [8] | SERREnable | This bit is the enable bit for the SERR# pin driver. A value of zero disables the SERR# driver. |

**PciStatus** This read/write register is used to record status information for PCI bus events.

Although this register is writable, write operations work in an unusual manner. Read/write bits in the register can be reset, but not set, by writing to this register. Bits are reset by writing a one to that bit position.

**PciStatus Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 0 | 0 | | 0 | 0 | 0 | 0 |

**PciStatus Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [4] | capabilitiesList | This read-only bit is always set, indicating the existence of a list of extended capabilities registers. The CapPtr register points to the start of the list. |

**PciStatus Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [7] | fastBackToBack | 3C90x NICs: This read-only bit indicates that the NIC, as a target, supports fast back-to-back transactions as defined in section 3.4.2 of the *PCI Specification*, revision 2.0.<br><br>3C90xB NICs with the 40-0502-00x ASIC: This bit is programmable through bit 0 in the EEPROM PciParm word. It must be programmed to zero to be compatible with other NICs in the 3C90x family.<br><br>3C900B NICs and 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC: This bit always returns zero. |
| [8] | dataParityDetected | The NIC sets this bit when, as a master, it detects the PERR# signal asserted, and the parityErrorResponse bit is set in the PciCommand register. |
| [10:9] | devselTiming | This read-only field is used to encode the slowest time with which the NIC asserts the DEVSEL# signal.<br><br>The NIC returns 01b, indicating support of "medium" speed DEVSEL# assertion. |
| [11] | signaledTargetAbort | The NIC asserts this bit when it terminates a bus transaction with target-abort. |
| [12] | receivedTargetAbort | The NIC asserts this bit when, operating as a bus master, its bus transaction is terminated with target-abort. |
| [13] | receivedMasterAbort | The NIC asserts this bit when, operating as a bus master, its bus transaction is terminated with master-abort. |
| [14] | signaledSystemError | This bit is set whenever the NIC asserts SERR#. |
| [15] | detectedParityError | The NIC asserts this bit when it detects a parity error, regardless of whether parity error handling is enabled. |

(2 of 2)

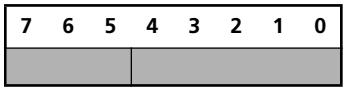
## RevisionId

The RevisionId register applies to 3C90xB NICs only.

This register provides a revision code for the ASIC. The first versions of the 3C90xB NICs ASICs return 00h.

The 3C900B NIC RegisterId format is undefined as of this writing.

**3C90xB NICs RevisionId Register Format**




**RevisionId Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [4:0] | rrevision | This field encodes the chip revision.<br><br>■ 3C90xB NICs with the 40-0502-00x ASIC: The entire field is used. Values may range from 00h to 1fh.<br><br>■ 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC: Bits [1:0] are hard-wired to 00b and bits [4:2] are used to encode the revision (this allows the pertinent revision data to also be visible in the RxOverruns register, as described in the next section) |

(1 of 2)

**RevisionId Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [7:5] | chip/Vendor | This field encodes the type of ASIC and the ASIC vendor:<br>■ 000 = 3C90xB NICs with the 40-0502-00x ASIC<br>■ 001 = 3C90xB NICs with the 40-0483-00x ASIC<br>■ 011 = 3C90xB NICs with the 40-0476-001 ASIC<br>Others = Reserved |

(2 of 2)

A portion of the RevisionId register value is also made visible in the I/O register space, in the asicRevision field in the NetworkDiagnostic register. The asicRevision field is only 5 bits wide, so only a portion of RevisionId can be visible there.

- For 3C900B NICs and 3C90xB NICs with the 40-0502-00x ASIC, bits [4:0] of RevisionId are visible in asicRevision.
- For 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC, bits [6:2] are visible in asicRevision, to allow both revision and vendor information to be visible here.

**ClassCode** This 24-bit read-only register identifies the general function of the PCI device. The NIC returns 020000h, indicating "Ethernet" network controller.

**CacheLineSize** This register applies to 3C90xB NICs only.

The system BIOS writes the system's cache line size into this register. The NIC uses this to optimize PCI bus master operation (choosing the best memory command, and so forth).

**CacheLineSize Register Format**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | | | | | | 0 | 0 |

The value in CacheLineSize represents the number of dwords in a cache. CacheLineSize only supports powers of 2 from 4 to 64 (giving a range of 16 to 256 bytes). An unsupported value is treated the same as zero.

**LatencyTimer** This 8-bit read/write register specifies, in units of PCI bus clocks, the value of the latency timer for bus master operations.

**LatencyTimer Register Format**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | 0 | 0 | 0 |

The system writes a value into LatencyTimer, which determines how long the NIC can hold the bus in the presence of other bus requesters. Whenever the NIC asserts FRAME#, the latency timer is started. When the timer count expires, the NIC must relinquish the bus as soon as its GNT# signal has been negated.

Because the low-order three bits are not implemented, the granularity of the timer is eight bus clocks.

**HeaderType** The value returned in this read-only field, 00h, identifies the NIC as a single-function PCI device, and specifies the configuration register layout shown in Table 14 and Table 15.

**IoBaseAddress** This read/write register allows the system to define the I/O base address for the NIC. PCI requires that I/O base addresses be set as if the system used 32-bit I/O addressing. The register returns one in bit 0 to indicate that this is an I/O base address.

- For 3C90x NICs, the upper 26 bits of the register are writable, indicating that the NIC requires 64 bytes of I/O space in the system I/O map.
- For 3C90xB NICs, the upper 25 bits of the register are writable, indicating that the NIC requires 128 bytes of I/O space in the system I/O map.

**3C90x NICs IoBaseAddress Register Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 |

**3C90x NICs IoBaseAddress Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [31:6] | ioBaseAddress | The system programs the I/O base address into this field. |

**3C90xB NICs IoBaseAddress Register Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**3C90xB IoBaseAddress Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [31:7] | ioBaseAddress | The system programs the I/O base address into this field. Because the NIC uses 128 bytes of I/O space, 25 bits are required to completely specify the I/O base address. |

**MemBaseAddress** This register applies to 3C90xB NICs only.

This read/write register allows the system to define the memory base address for the NIC's registers. The register returns zero in bit 0 to indicate that this is a memory base address.

The upper 25 bits of the register are writable, indicating that the NIC requires 128 bytes of memory space in the system I/O map.

**MemBaseAddress Register Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | | 0 |

**MemBaseAddress Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [1] | mapLowerMeg | This read-only bit, when set, instructs the system to map the NIC registers into the lowest 1 MB of memory address space. This bit is loaded from the PciParm word in EEPROM. |
| [31:7] | memBaseAddress | The system programs the memory base address into this field. Because the NIC registers occupy 128 bytes of space, 25 bits are required to completely specify the memory base address. |

**SubsystemVendorId** This register applies to 3C90xB NICs only. This value is read from EEPROM location 17h after system reset.

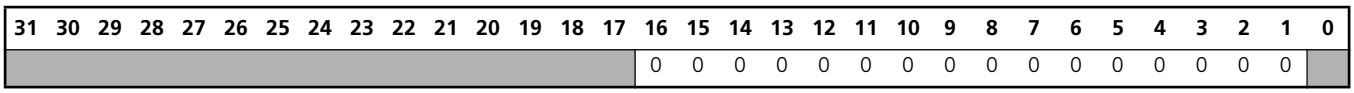
For more information, see "SubsystemVendorId" in Chapter 5.

**SubsystemId** This register applies to 3C90xB NICs only. This is the value read from EEPROM word 18h after system reset.

For more information, see "SubsystemId" in Chapter 5.

**BiosRomControl** This read/write register allows the system to define the base address for the NIC's BIOS ROM.

**BiosRomControl Register Format**




**BiosRomControl Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | addressDecodeEnable | When this bit is cleared, the NIC's BIOS ROM is disabled. Setting this bit when the memorySpace bit in the PciCommand register is also set causes the NIC to respond to accesses in its configured expansion ROM space. |
| [31:17] | romBaseAddress | The system programs the expansion ROM base address into this field.<br>Since this field is 15 bits wide, the ROM is mapped on 128 KB boundaries. If a ROM smaller than 128 KB is installed, it will appear as multiple images within the 128 KB space. |

**CapPtr** This register applies to 3C90xB NICs only.

This hard-coded value points to the beginning of a chain of registers that describe enhanced functions. This register returns dch, which points to the power management registers.

**InterruptLine** This 8-bit read/write register is written by the system to communicate to the device driver which interrupt level is being used for the device. This allows the driver to use the appropriate interrupt vector for its interrupt service routine (ISR).

For 80x86 systems, the value in InterruptLine corresponds to the IRQ numbers (0 through 15) of the standard dual 8259 configuration, and the values 0 and 255 correspond to *disabled*.

**InterruptPin** This read-only register indicates which PCI interrupt "pin" the NIC will use.

The 3C90xB NICs always use INTA#, so 01h is returned in InterruptPin.

**MinGnt** MinGnt is a read-only value that specifies, in 250-ns increments, how long a burst period the NIC requires when it is operating as a bus master. The value for MinGnt is stored in the PciParm word in EEPROM.

- 3C90x NICs return the value 3 in this field.
- 3C90xB NICs return the value 10d in this field.

These values assume a 33 MHz bus (30-ns clock period), 1 clock for address phase, 10 clock latency to first data phase, then no wait states for the 64 remaining data phases, as illustrated by the following formula:

```
(1 + 10 + 64)30 ns = 2.25 µs, rounded up to 2.5 µs; 2.5 µs/250 ns = 10
```

**MaxLat** MaxLat is a read-only value that specifies, in 250-ns increments, how often the NIC requires the bus when operating as a bus master. The value for MaxLat is stored in the PciParm word in EEPROM.

- 3C90x NICs return a value of 8 in this field, implying a latency tolerance of 2 µs. The NIC needs to transfer 64 bytes (assuming a 16 dword burst length).
- 3C90xB 100 Mbps NICs return the value 10d in this field, implying a latency tolerance of 2.5 µs. This assumes that the NIC operates in a system that allows 64-byte maximum bursts and full-duplex operation. See the following formula:

```
64 bytes/(12.5 MBps x 2) = 2.56 µs; round down to 2.5 µs; 2.5 µs/250 ns = 10d
```

- 3C90xB 10 Mbps-only NICs return the value 48d in this field, implying a latency tolerance of 12 µs. This assumes that the NIC operates in a system that allows 32-byte maximum bursts and full-duplex operation. See the following formula:

```
32 bytes/(1.25 MBps x 2) = 12.8 µs; round down to 12.5 µs; 12.5 µs/250 ns = 50d; round down to
48d for a more even binary representation
```

## Power Management

This section describes the power management registers. For details on power management operation, see "Power Management" in Chapter 3.

*The power management registers apply to the 3C90xB NICs only.*

**CapID** This byte-wide, read-only register indicates the type of capability data structure. It returns 01h to indicate a PCI power management structure.

**NextPtr** This byte-wide, read-only register points to the next capability data structure in the capabilities list. It returns 00h to indicate that there are no further data structures.

**PowerMgmtCap** This read-only register provides information about the NIC's power management capabilities. The reset default is 7601h, but several bits are loaded from EEPROM shortly after reset.

**PowerMgmtCap Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | | | |

**PowerMgmtCap Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [2:0] | version | This read-only field returns 1h, as specified in the *PCI Bus Power Management Specification*. |
| [9] | d1Support | This read-only bit, when set, indicates that this device supports the D1 power state.<br>This value of this bit is determined by bit 4 (d1Support) in the EEPROM PciParm word. |
| [10] | d2Support | This read-only bit, when set, indicates that this device supports the D2 power state.<br>This value of this bit is determined by bit 5 (d2Support) in the EEPROM PciParm word. |
| [15:11] | pmeSupport | This read-only field indicates the power states from which this device is able to generate a power management event (assert PME#).<br>Each bit corresponds to a power state. A zero in a particular bit indicates that events cannot be generated from that state. The bits are defined as follows:<br>■ xxxx1: Power management events possible from D0.<br>■ xxx1x: Power management events possible from D1.<br>■ xx1xx: Power management events possible from D2.<br>■ x1xxx: Power management events possible from $D3_{hot}$.<br>■ 1xxxx: Power management events possible from $D3_{cold}$. |

The 3C90xB-TX NICs with the 40-0502-00x ASIC and the 3C900B-FL and 3C905B-FX NICs hard-wire bit 11 to 0 and bit 14 to 1. The values of bits 12, 13, and 15 are determined by bits 4, 5, and 3 (respectively) in the EEPROM PciParm word.

The 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC and the 3C900B and later NICs hard-wire bit 11 to 0. The values of bits 12, 13, 14, and 15 are determined by bits 4, 5, 0, and 3 (respectively) in the EEPROM PciParm word.

**PowerMgmtCtrl** This read/write register allows control over the power state and the power management interrupts.

*PowerMgmtCtrl is also mapped into the normal register space, I/O or memory mapped, at offset 7c. In this location, PowerMgmtCtrl is write-only, and only bits [1:0] and 8 are writable (pmeStatus cannot be written).*

The reset default is 0000h.

**PowerMgmtCtrl Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | | |

**PowerMgmtCtrl Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [1:0] | powerState | This read/write field is used to determine or set the NIC's power state. The following values are defined:<br>■ 0 = State D0<br>■ 1 = State D1<br>■ 2 = State D2<br>■ 3 = State D3<br>If this bit is set to a nonzero value, the NIC does not respond to PCI I/O or memory cycles, nor is it able to generate PCI bus master cycles. |
| [8] | pmeEn | When this read/write bit is set, the NIC is allowed to report wake-up events on the PME# signal. The specific events that can generate wake-up are defined by the PowerMgmtEvent I/O register. |
| [15] | pmeStatus | This read/clear bit is set to indicate that a wake-up event has occurred. This bit is set regardless of the value in the pmeEn bit. Writing a one to this bit clears it. Writing zero has no effect. |

## PowerMgmtEvent

| | |
|---|---|
| Synopsis | Allows control over power management event generation. |
| Type | Read/write |
| Size | 16 bits |
| Window | 7 |
| Offset | c |

The PowerMgmtEvent register contains enable bits to control which types of events can generate a wake-up event to the host system. It also contains status bits that indicate what specific events have occurred.

PowerMgmtEvent is cleared by reset.

**PowerMgmtEvent Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | 0 | | | |

Bits [2:0] determine what kinds of events can cause the NIC to generate a wake-up event (interrupt) on the PCI bus. All three bits are qualified with the pmeEn bit in the PowerMgmtCtrl configuration register: if pmeEn is clear, then wake-up generation is disabled and these bits are ignored. Also, these bits have no effect in D0.

**PowerMgmtEvent Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | wakeupPktEnable | Setting this read/write bit enables the NIC to generate wake-up events caused by wake-up packet reception. |
| [1] | magicPktEnable | Setting this read/write bit enables the NIC to generate wake-up events caused by Magic Packet reception. |

(1 of 2)

**PowerMgmtEvent Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [2] | linkEventEnable | Setting this read/write bit enables the NIC to generate wake-up events caused by a change in link status (network cable connect or disconnect). |
| Bits [6:4] indicate that an actual wake-up event has occurred. These bits are masked by their corresponding enable bits [2:0]. If the enable bit is not set, then the event bit can never become set. Once set, these bits are cleared by a read to PowerMgmtEvent. | | |
| [4] | wakeupPktEvent | Indicates that a wake-up packet that meets the reception criteria set by software has been received. |
| [5] | magicPktEvent | Indicates that a Magic Packet frame has been received. |
| [6] | linkEvent | Indicates that a link status event has occurred. |

(2 of 2)

# 5 EEPROM

This chapter provides information about the EEPROM contents and registers.

## Data Format

Table 16 summarizes the contents of the 3C90x NICs EEPROM.

Table 17 summarizes the contents of the 3C90xB NICs EEPROM.

**Table 16** 3C90x NICs EEPROM Contents

| Offset (hex) | Field Name | 10/100 TX (hex) | 10-only TPO (hex) | 10-only COMBO (hex) |
|---|---|---|---|---|
| 00 | 3Com Node Address (word 0) | xxxx | xxxx | xxxx |
| 01 | 3Com Node Address (word 1) | xxxx | xxxx | xxxx |
| 02 | 3Com Node Address (word 2) | xxxx | xxxx | xxxx |
| 03 | DeviceId | 9050 | 9000 | 9001 |
| 04 | Manufacturing Data - Date | xxxx | xxxx | xxxx |
| 05 | Manufacturing Data - Division | 00xx | 00xx | 00xx |
| 06 | Manufacturing Data - Product Code | xxxx | xxxx | xxxx |
| 07 | ManufacturerId | 6d50 | 6d50 | 6d50 |
| 08 | PciParm | 0418 | 0418 | 0418 |
| 09 | RomInfo | 0000 | 0000 | 0000 |
| 0a | OEM Node Address (word 0) | xxxx | xxxx | xxxx |
| 0b | OEM Node Address (word 1) | xxxx | xxxx | xxxx |
| 0c | OEM Node Address (word 2) | xxxx | xxxx | xxxx |
| 0d | Software Information | 0010 | 0010 | 0010 |
| 0e | Compatibility Word | 0000 | 0000 | 0000 |
| 0f | Software Information 2 | 0000 | 0000 | 0000 |
| 10 | CapabilitiesWord | 10a6 | 00a6 | 00a6 |
| 11 | Reserved | 0000 | 0000 | 0000 |
| 12 | InternalConfig Word 0 | 02d8 | 02d8 | 02d8 |
| 13 | InternalConfig Word 1 | 0163 | 0103 | 0103 |
| 14 | Reserved | 0000 | 0000 | 0000 |
| 15 | Software Information 3 | 0002 | 0000 | 0000 |
| 16 | Reserved | 0000 | 0000 | 0000 |
| 17 | Checksum | 00xx | 00xx | 00xx |

**Table 17** 3C90xB NICs EEPROM Contents

| Offset (hex) | Field Name | 10/100 TX (hex) | 10-only TPO (hex) | 10-only COMBO (hex) | 10-only TPC (hex) |
|---|---|---|---|---|---|
| 00 | 3Com Node Address (word 0) | xxxx | xxxx | xxxx | xxxx |
| 01 | 3Com Node Address (word 1) | xxxx | xxxx | xxxx | xxxx |
| 02 | 3Com Node Address (word 2) | xxxx | xxxx | xxxx | xxxx |
| 03 | DeviceId | 9055 | 9004 | 9005 | 9006 |
| 04 | Manufacturing Data - Date | xxxx | xxxx | xxxx | xxxx |
| 05 | Manufacturing Data - Division | 00xx | 00xx | 00xx | 00xx |
| 06 | Manufacturing Data - Product Code | xxxx | xxxx | xxxx | xxxx |
| 07 | ManufacturerId | 6d50 | 6d50 | 6d50 | 6d50 |
| 08 | PciParm | 2970 (40-0502-00x ASIC)<br>2979 (40-0476-001 or 40-0483-00x ASIC) | c171 | c171 | c171 |
| 09 | RomInfo | 0000 | 0000 | 0000 | 0000 |
| 0a | OEM Node Address (word 0) | xxxx | xxxx | xxxx | xxxx |
| 0b | OEM Node Address (word 1) | xxxx | xxxx | xxxx | xxxx |
| 0c | OEM Node Address (word 2) | xxxx | xxxx | xxxx | xxxx |
| 0d | Software Information | 0010 | 0010 | 0010 | 0010 |
| 0e | Compatibility Word | 0000 | 0000 | 0000 | 0000 |
| 0f | Software Information 2 | 0000 | 0000 | 0000 | 0000 |
| 10 | CapabilitiesWord | 32a2 | 22a2 | 22a2 | 22a2 |
| 11 | Reserved | 0000 | 0000 | 0000 | 0000 |
| 12 | InternalConfig Word 0 | 0000 | 0000 | 0000 | 0000 |
| 13 | InternalConfig Word 1 | 0180 | 0180 | 0180 | 0180 |
| 14 | Analog Diagnostic | 0000 | 0000 | 0000 | 0000 |
| 15 | Software Information 3 | 0000 | 0000 | 0000 | 0000 |
| 16 | LANWorks Data | 0000 | 0000 | 0000 | 0000 |
| 17 | SubsystemVendorId | 10b7 | 10b7 | 10b7 | 10b7 |
| 18 | SubsystemId | 9055 | 9004 | 9005 | 9006 |
| 19 | MediaOptions | 000a | 0008 | 0038 | 0018 |
| 1a–1f | Reserved | 0000 | 0000 | 0000 | 0000 |
| 20 | Checksum | 00xx | 00xx | 00xx | 00xx |

## 3Com Node Address

This field contains the 3Com node address for the NIC. This is *not* the field to be programmed into the StationAddress register. See "OEM Node Address" later in this chapter.

## DeviceId

This field contains the 2-byte product identifier, which gets loaded into the ASIC and made available in the DeviceId register in the PCI Configuration space.

The most-significant three nibbles are the numeric portion of the 3C number:

- 905 for 10/100 Mbps PCI NICs
- 900 for 10 Mbps-only PCI NICs

The least-significant nibble is used as a sort of revision code, to reflect the particular transceiver resources on the NIC and, potentially, board or ASIC revisions.

The following codes are defined:

**3C90x NICs**

| | |
|---|---|
| 9000 | PCI 10/100 Mbps; shared 10BASE-T/100BASE-TX connector |
| 9001 | PCI 10/100 Mbps; shared 10BASE-T/100BASE-T4 connector |
| 9050 | PCI 10BASE-T (TPO) |
| 9051 | PCI 10BASE-T/10BASE2/AUI (COMBO) |

**3C90xB NICs**

| | |
|---|---|
| 9055 | PCI 10/100 Mbps; shared 10BASE-T/100BASE-TX connector |
| 9056 | PCI 10/100 Mbps; shared 10BASE-T/100BASE-T4 connector |
| 9004 | PCI 10BASE-T (TPO) |
| 9005 | PCI 10BASE-T/10BASE2/AUI (COMBO) |
| 9006 | PCI 10BASE-T/10BASE2 (TPC) |
| 900A | PCI 10BASE-FL |
| 905A | PCI 10BASE-FX |

## Manufacturing Data

The manufacturing data fields are described below.

### Date

This field contains the date of manufacture, encoded as follows:

| | |
|---|---|
| Day | [4:0]: The day (1 through 31) |
| Month | [8:5]: The number of the month (1 through 12) |
| Year | [15:9]: The last two digits of the current year (0 through 99) |

### Division

This field contains the manufacturing division code, as shown on the product bar code label.

### Product Code

This field contains the manufacturing product code, which is two alphanumeric ASCII characters from the bar code label.

## ManufacturerId

This field contains 3Com's assigned EISA Manufacturer ID. It is a byte-swapped, encoded form of the string "TCM." This is included to aid software in identifying 3Com NICs in systems where a PCI BIOS is not available.

This value has no significance in PCI operation (it is unrelated to the PCI VendorId value). It is not used by the NIC logic in any way nor made available in any NIC I/O register. This value is the same as the value in this EEPROM location in 3Com ISA/EISA NICs.

## RomInfo

This field conveys to a driver or configuration program whether a BIOS ROM is installed, and the physical size of the ROM.

**RomInfo Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [11] | romPresent | Indicates the presence of a BIOS ROM. |
| [13:12] | romSize | The physical size of the BIOS ROM. The romSize bit is valid only when the romPresent bit is set.<br>■ 00 = 64K × 8<br>■ 01 = 128K × 8<br>■ 1x = Reserved |

## PciParm

The contents of this field are loaded into the ASIC to control various hardware functions related to PCI bus operation.

**PciParm Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | fastBackToBack | This bit applies to 3C90x and 3C90xB NICs with the 40-0502-00x ASIC only.<br>■ 3C90x NICs: This bit determines the value for the fastBackToBack bit 7 in the PciStatus register.<br>■ 3C90xB NICs with the 40-0502-00x ASIC: This bit determines the value for the fastBackToBack bit 7 in the PciStatus register. This bit must always be programmed to zero. |
| [0] | d3Hot | This bit applies to 3C900B and 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC only.<br>This bit determines the value for bit 14 in the PowerMgmtCap register. This bit indicates the NIC's ability to generate power management events from the $D3_{hot}$ state. |
| [1] | lower1Meg | This bit provides the value for the mapLowerMeg bit in the MemBaseAddress register. |
| [2] | disableMemBase | When set, disables the MemBaseAddress register. This bit causes the memBaseAddress bit to read back as zeros, appearing as if it is not implemented by the NIC. |
| [3] | d3ColdPme | Provides the value returned in bit 15 of the PowerMgmtCap register. This bit, when set, indicates that the NIC is capable of signaling wake-up from the $D3_{cold}$ state. |

(1 of 2)

**PciParm Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [4] | d1Support | Provides the value returned in the d1Support bit and also bit 12 in the PowerMgmtCap register. |
| [5] | d2Support | Provides the value returned in the d2Support bit and also bit 13 in the PowerMgmtCap register. |
| [9:6] | minGnt | Determines the value returned in bits [4:1] of the MinGnt register. |
| [15:10] | maxLat | Determines the value returned in bits [5:0] of the MaxLat register. |

(2 of 2)

PciParm has a default value of 1570h but is normally overwritten by the value in EEPROM.

The EEPROM values specify the following:

| Bit | 3C90x and 3C90xB (with 40-0502-00x ASIC) 10/100 Mbps NICs | 3C90xB (with 40-0476-001 or 40-0483-00x ASIC) 10/100Mbps NICs | 3C90x 10 Mbps NICs | 3C900B 10 Mbps NICs |
|---|---|---|---|---|
| d3Hot | N/A | 1 | N/A | 1 |
| fastBackToBack | 0 | N/A | 0 | N/A |
| lower1Meg | 0 | 0 | 0 | 0 |
| disableMemBase | 0 | 0 | 0 | 0 |
| d3ColdPme | 0 | 0 | 0 | 0 |
| d1Support | 1 | 1 | 1 | 1 |
| d2Support | 1 | 1 | 1 | 1 |
| minGnt | 0101b (2.5 µs) | 0101b (2.5 µs) | 0101b (2.5 µs) | 0101b (2.5 µs) |
| maxLat | 001010b (2.5 µs) | 001010b (2.5 µs) | 110000b (12 µs) | 110000b (12 µs) |

## OEM Node Address

This is the field to be programmed into the StationAddress register. For 3Com NICs, this field contains the same value as in 3Com Node Address. OEM customers may choose to program this field with a different value.

## Software Information

This field contains environmental information for use by the driver.

**Software Information Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [3:0] | Reserved | Reserved, set to zero. |
| [5:4] | optimizeFor | Specifies the environment for which to optimize.<br>■ 00 = Reserved<br>■ 01 = Normal<br>■ 10 = Maximum network performance<br>■ 11 = Minimum CPU utilization |
| [7:6] | Reserved | Reserved, set to zero. |

(1 of 2)

**Software Information Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [13:8] | Reserved | Reserved. The value of these bits is undefined. |
| [14] | linkBeatDisable | Indicates to the host software whether it should set the linkBeatEnable bit in the MediaStatus register (for appropriately equipped NICs).<br>Note the opposite polarities of the linkBeatDisable and linkBeatEnable bits. |
| [15] | fullDuplex | Indicates to the host software whether it should configure the NIC for full-duplex operation.<br>■ 0 = Disable full-duplex<br>■ 1 = Enable full-duplex |

(2 of 2)

The default value specifies:

- Normal optimization
- Link beat enabled
- Full-duplex disabled

## Compatibility Word

This field contains two byte-wide values that are checked by the driver with an internal value (CLevel) to determine the compatibility of the driver with the software.

**Compatibility Word Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [7:0] | warningLevel | If the driver's CLevel is less than this field, the driver issues a warning message that a newer driver is available that may offer improved performance. |
| [15:8] | failureLevel | If the driver's CLevel is less than this field, the driver fails the installation process. A new driver needs to be obtained. |

## Capabilities Word

This word contains data defining the basic capabilities of the NICs.

Table 18 summarizes the capabilities of 3C90x NICs.

Table 19 summarizes the capabilities of 3C90xB NICs.

**Table 18** 3C90x NICs Summary of Capabilities

| Bit | Capabilities Bit | Value |
|---|---|---|
| 0 | supportsPlugNPlay | 0 |
| 1 | supportsFullDuplex | 1 |
| 2 | supportsLargePackets | 1 |
| 3 | supportsSlaveDma | 0 |
| 4 | supportsSecondDma | 0 |

(1 of 2)

**Table 18** 3C90x NICs Summary of Capabilities (continued)

| Bit | Capabilities Bit | Value |
|---|---|---|
| 5 | supportsFullBusMaster | 1 |
| 6 | supportsFragBusMaster | 0 |
| 7 | supportsCrcPassThru | 1 |
| 8 | supportsTxDone | 0 |
| 9 | supportsNoTxLength | 0 |
| 10 | supportsRxRepeat | 0 |
| 11 | supportsSnooping | 0 |
| 12 | supports100Mbps | 1 |
| 13 | supportsPowerMgmt | 0 |

(2 of 2)

For 3C90x NICs, the resulting values are:

- 10A6h (10/100 Mbps NICs)
- 00A6h (10 Mbps-only NICs)

**Table 19** 3C90xB NICs Summary of Capabilities

| Bit | Capabilities Bit | Value |
|---|---|---|
| 0 | supportsPlugNPlay | 0 |
| 1 | supportsFullDuplex | 1 |
| 2 | supportsLargePackets | 0 |
| 3 | supportsSlaveDma | 0 |
| 4 | supportsSecondDma | 0 |
| 5 | supportsFullBusMaster | 1 |
| 6 | supportsFragBusMaster | 0 |
| 7 | supportsCrcPassThru | 1 |
| 8 | supportsTxDone | 0 |
| 9 | supportsNoTxLength | 1 |
| 10 | supportsRxRepeat | 0 |
| 11 | supportsSnooping | 0 |
| 12 | supports100Mbps | 0 or 1 |
| 13 | supportsPowerMgmt | 1 |

For 3C90xB NICs, the resulting values are:

- 32A2h (10/100 Mbps NICs)
- 22A2h (10 Mbps-only NICs)

The capabilities bits that are set for the NICs are described below.

**Capabilities Word Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [1] | supportsFullDuplex | Indicates that the NIC supports full-duplex media operation. |
| [2] | supportsLargePackets | Indicates that the NIC supports packet sizes over 2047 bytes (3C90x NICs only). |
| [6] | supportsFullBusMaster | Indicates that the NIC supports scatter/gather bus master data transfers. |
| [7] | supportsCrcPassThru | Indicates that the NIC supports CRC pass-through using the crcAppendDisable bit in the Frame Start Header (FSH). |
| [9] | supportsNoTxLength | Indicates that the NIC calculates the length of transmit packets automatically—software does not need to supply it in the FSH. |
| [12] | supports100Mbps | Indicates the NIC's ability to support 100 Mbps data rates. |
| [13] | supportsPowerMgmt | Indicates that the NIC supports the OnNow/ACPI power management scheme. |

## InternalConfig

Two words supply the value for the InternalConfig register:

- InternalConfig word 0 corresponds to the low-order 16 bits of the InternalConfig registers.
- InternalConfig word 1 corresponds to the high-order 16 bits.

The hardware reads the words automatically upon reset to provide default settings for non–system-related configuration settings. They can later be written over by driver software.

See "InternalConfig" in Chapter 4 for bit definitions.

## AnalogDiagnostic

This word is reserved for later versions of 3C900B NICs that implement on-chip transmit filtering.

## Software Information 2

This word contains additional information for drivers. For 3C90x NICs, there is no defined use for Software Information 2.

**3C90xB NICs Software Information 2 Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [1] | fixedBroadcastRxBug | Indicates that the bug that required a bit to be turned on in the multicast hash filter in order to receive broadcast frames has been fixed. |
| [2] | fixedEndecLpbackBug | Indicates that the bug in the ENDEC loopback function has been fixed. |
| [3] | wolConnector | Indicates that the NIC has a 3-pin Remote Wake-Up connector. |

(1 of 2)

**3C90xB NICs Software Information 2 Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [4] | pmePulsed | This bit is meaningless if the wolConnector bit is 0.<br>When wolConnector is 1, the value of this bit indicates the type of Remote Wake-Up PME signal:<br>■ 0 = Level signal<br>■ 1 = Pulsed signal (the Dell implementation) |
| [5] | fixedMWIBug | Indicates that the Memory Write Invalidate (MWI) PCI command bug has been fixed and the MWI command should be enabled.<br>■ 0 = MWI command is not working properly<br>■ 1 = MWI command is working properly |

(2 of 2)

## Software Information 3

This word instructs the driver how to configure the NIC when "forcing" the transceiver to MII or auto-negotiation.

**Software Information 3 Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [3:0] | forceXcvr | Specifies which transceiver type to select on an MII-based PHY device.<br>■ 0000 = Generic MII<br>■ 0001 = 100BASE-T4 - MII<br>■ 0010 = 10BASE-T - MII<br>■ 0011 = 100BASE-TX - MII<br>■ 0100 = 10BASE-T - Auto-negotiation<br>■ 0100 = 100BASE-TX - Auto-negotiation<br>■ Others = Reserved for future combinations<br>In addition to this field, driver software should check the fullDuplex bit in the Software Information field to determine which duplex mode to configure for the NIC.<br>This value is used by the driver only when both of the following conditions are true:<br>■ The autoSelect bit in the InternalConfig register is clear.<br>■ The xcvrSelect bit in the InternalConfig register is set to MII or auto-negotiation. |
| [15:4] | Reserved | Reserved for future information bits. |

## Lanworks Data

This word is reserved for use by Lanworks to hold data related to their expansion ROMs.

## SubsystemVendorId

This word applies to 3C90xB NICs only.

This word is the 2-byte subsystem vendor ID. Because in this case the subsystem is a 3Com NIC, 3Com's PCI vendor ID, 10b7h, is used.

## SubsystemId

This word applies to 3C90xB NICs only.

This is the 2-byte subsystem ID. For 3Com NICs, use the same code as the DeviceId data field.

## MediaOptions

This field holds the value to be loaded into the MediaOptions register after reset. For details, see "MediaOptions" in Chapter 12.

## Checksum

For 3C90x NICs, the checksum for the EEPROM contents is a byte-wide XOR computed across all bytes in EEPROM words zero through 16h, and written into the low-order byte of word 17h.

For 3C90xB NICs, the checksum for the EEPROM contents is a byte-wide XOR computed across all bytes in EEPROM words zero through 19h, and written into the low-order byte of word 20h.

## EepromCommand

| | |
|---|---|
| Synopsis | Allows commands to be issued to the serial EEPROM controller. |
| Type | Read/write |
| Size | 16 bits |
| Window | 0 |
| Offset | a |

**EepromCommand Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | |

**EepromCommand Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [5:0] | eepromAddress | These six read/write bits identify one of the 64 16-bit words to be the target for the ReadRegister, WriteRegister, and EraseRegister commands.<br>Bits 5 and 4 are further defined to identify an individual command among the following group of four subcommands:<br>■ 00 = WriteDisable (60 µs)<br>■ 01 = WriteAll (11 ms)<br>■ 10 = EraseAll (11 ms)<br>■ 11 = WriteEnable (60 µs)<br>The definition of bits 5 and 4 is valid when the eepromOpcode bits [7:6] equal $00_2$. |
| [7:6] | eepromOpcode | These two read/write bits specify one of three individual commands and a single group of four subcommands. The opcodes are:<br>■ 00 = Write Enable/Disable and Write/Erase All subcommands |

(1 of 2)

**EepromCommand Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| | | ■ 01 = WriteRegister (11 ms)<br>■ 10 = ReadRegister (162 µs)<br>■ 11 = EraseRegister (11 ms) |
| [15] | eepromBusy | This read-only bit is asserted during the execution of EEPROM commands. Further commands should not be issued to the EepromCommand register, nor should data be read from the EepromData register while this bit is true. |

(2 of 2)

The EepromCommand register provides the host with a method for controlling the NIC's serial EEPROM. Individual 16-bit word locations within the EEPROM may be written, read, or erased. Also, the EEPROM's WriteEnable, WriteDisable, EraseAll, and WriteAll commands can be issued.

Two-bit opcodes and 6-bit addresses are written to the least-significant eight bits of this register to cause the NIC to carry out the desired EEPROM command. If data is written to the EEPROM, the host must write the 16-bit data word to the EepromData register before issuing the associated write command. Similarly, if data is to be read from the EEPROM, the read data is available through EepromData 162 µs after the ReadRegister command has been issued.

A mechanism within the EEPROM interface automatically disables writes and erasures to prevent accidental data changes should power be interrupted. The NIC disables writes and erasures after every write or erase type command has been executed. To write or erase a series of locations, the host must issue the WriteEnable command before every write or erase type command.

The serial EEPROM can only clear bits to zero during a write command and cannot set individual bits to ones. Therefore, an EraseRegister or EraseAll command must be issued before data is written to the EEPROM.

The EEPROM is a particularly slow device. It is important that the host wait until the eepromBusy bit is false before it issues a command to the EepromCommand register.

A typical write operation would be controlled as follows:

1. Verify that the eepromBusy bit is false.
2. Issue a WriteEnable command; opcode = 0011 $XXXX_2$.
3. Verify that eepromBusy is false.
4. Issue an EraseRegister command; opcode = 11aa $aaaa_2$.
5. Verify that eepromBusy is false.
6. Write the data pattern to the EepromData register.
7. Issue a WriteEnable command; opcode = 0011 $XXXX_2$.
8. Verify that eepromBusy is false.
9. Issue a WriteRegister command; opcode = 01aa $aaaa_2$.
10. The EepromCommand register defaults to 0000h upon reset.

## EepromData

| | |
|---|---|
| Synopsis | Provides data access for the EEPROM. |
| Type | Read/write |
| Size | 16 bits |
| Window | 0 |
| Offset | c |

The EepromData register is a 16-bit port for use with the NIC's serial EEPROM. The host can read data out of the EEPROM from this register when the eepromBusy bit becomes false. Data to be written to the EEPROM is written to this register before the write command is issued to the EepromCommand register.

EepromData is cleared after a system reset.

# 6 DOWNLOAD AND TRANSMISSION

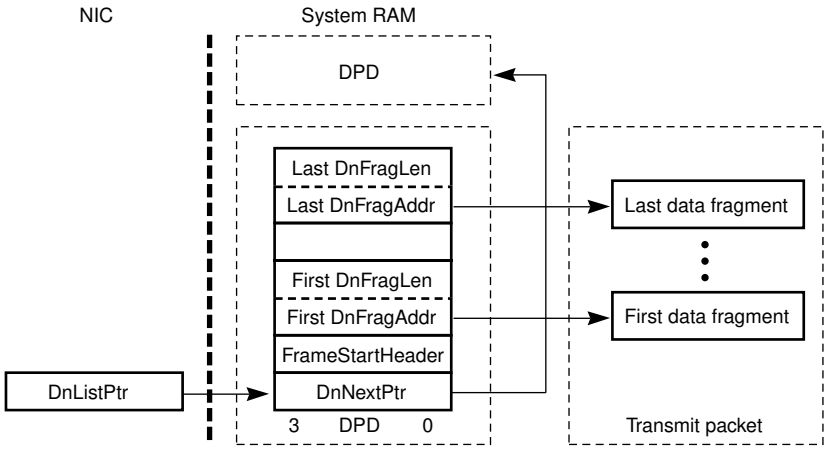
This chapter presents an overview of the packet download and transmission process, and defines the registers associated with the downloading and transmission of data.

The NICs support a multipacket, multifragment gather process, whereby descriptors representing packets can be built in system memory and linked together by the host. The NIC follow the links, downloading multiple fragments per packet, and generating interrupts when required.

## Packet Download Model

Drivers control packet download by building a linked list of packet descriptors, called down packet descriptors (DPDs). This linked list, which is called the downlist, is illustrated in Figure 25.

**Figure 25** Downlist




The packet to be transmitted is first placed in data fragments (buffers) in system memory. Next, a list of DPDs (the downlist) that points to the fragments is also created in system memory.

The head of the list is the DPD that corresponds to the current download packet. The down list pointer (DnListPtr) register points to this DPD. As the DPD is processed, the fragment address and fragment length values are fetched one by one from the DPD into on-NIC registers, which are used to control the data download operations.

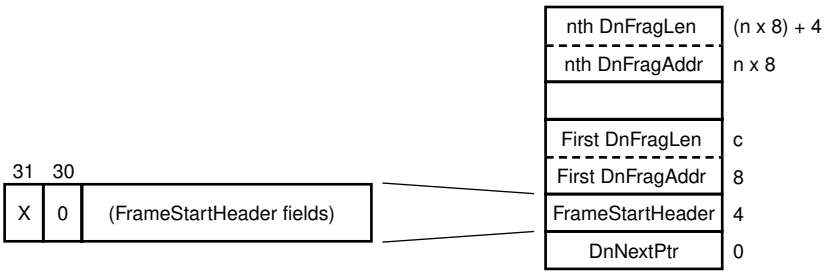
The NIC exits reset with the download engine in the idle state, ready to start processing a downlist as soon as a nonzero value is written into DnListPtr.

## DPD Data Structure

Two DPD formats are supported: Type 0 and Type 1.

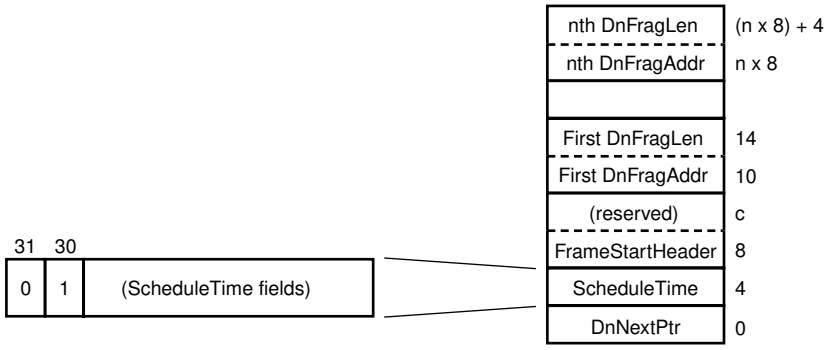
The Type 0 DPD format is shown in Figure 26. This format is backward-compatible with 3C90x NICs but has some added extensions.

**Figure 26** Type 0 DPD Format




The Type 1 DPD format is shown in Figure 27. This format adds fields to support time scheduling of packet downloads.

**Figure 27** Type 1 DPD Format




DPDs are between 16 and 512 bytes long and describe up to 63 fragments.

Bits [31:30] at offset 4 determine whether the DPD format is Type 0 or Type 1. Bit combination 11b is reserved for a future DPD format. The bit combinations are summarized in Table 20.

**Table 20** DPD Format Bit Combinations

| Bit 31 | Bit 30 | Backward-Compatible with 3C90x NICs? |
|---|---|---|
| 0 | 0 | Yes |
| 0 | 1 | No |
| 1 | 0 | Yes |
| 1 | 1 | Reserved |

### Down Next Pointer

The first dword in the DPD is the DnNextPtr entry, which contains the physical address of the next DPD in the downlist. If there are no more DPD entries in the downlist, then this value is zero.

**Type 0 DnNextPtr Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 |

**Type 1 DnNextPtr Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |

Type 0 DPDs must be aligned on 8-byte physical address boundaries.

Type 1 DPDs must be aligned on 16-byte boundaries.

When the download engine idles because it is reading a zero DnNextPtr DPD entry, the NIC can poll on DnNextPtr and wait for the software to write a nonzero value to it. This function is enabled when the DnPoll register contains a nonzero value.

### Frame Start Header

The FrameStartHeader DPD entry (also called the FSH) contains packet control information.

The FSH entry differs for the 3C90x and 3C90xB NICs. Each entry is described below.

**3C90x NICs FrameStartHeader Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | | | | | | | | | | | | | | |

**3C90x NICs FrameStartHeader Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [12:0] | txLength | This field contains the total length of the packet. Normally, the value in this field equals the sum of the dnFragLen fields in the DPD.<br><br>The NIC pads packets to the 60-byte minimum, so this field should not be set to 60 unless that number of bytes of data is actually available to download. |
| [13] | crcAppendDisable | The driver sets this bit to inhibit the NIC from appending a CRC to the end of this packet.<br><br>When this bit is set, it is expected that the packet's CRC would be supplied as part of the data downloaded to the FIFO. An exception to this occurs with a transmit underrun. In this case a guaranteed-bad CRC is appended to the packet.<br><br>When this bit is cleared, the NIC computes and appends CRCs for transmit packets. |

(1 of 2)

**3C90x NICs FrameStartHeader Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [15] | txIndicate | When this bit is set, a txComplete interrupt occurs when a packet finishes transmitting. If this bit is cleared, no interrupt occurs unless a transmit error occurs. |
| [31] | dnIndicate | When this bit is set, a dnComplete interrupt occurs when a packet finishes downloading. If this bit is cleared, no interrupt occurs.<br>The NIC reads this bit after the download operation has finished, allowing the host to change this bit while the download is in progress.<br>The FSH is read twice: first to store the FSH bit values temporarily while the packet is being downloaded, and again after the download is finished to determine whether to generate an interrupt. |

(2 of 2)

**3C90xB NICs FrameStartHeader Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | 0 | | 0 | 0 | 0 | | | | | | | | | | |

**3C90xB NICs FrameStartHeader Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [1:0] | rndupBndry | These bits determine the boundary to which transmit packet lengths are rounded up in the transmit FIFO, and hence onto the network medium. These bits are ignored if rndupDefeat is set.<br>■ 00 = Up to dword<br>■ 10 = Up to word<br>■ x1 = No rounding up is performed<br>In 3C90x NICs, these bits were part of the txLength [12:0] field, which 3C90xB NICs no longer require. |
| [9:2] | pktId | This field can be used as a packet ID or sequence number. This value is saved with the packet in the transmit FIFO, and made visible in the TxPktId register while the packet is being transmitted. When a transmit error occurs, the driver can check TxPktId to determine which packet experienced the error.<br>In 3C90x NICs, these bits were part of the txLength [12:0] field, which 3C90xB NICs no longer require.<br>3C90xB NICs ignore the length value that drivers written for 3C90x NICs write in this field. |
| [13] | crcAppendDisable | The driver sets this bit to inhibit the NIC from appending a CRC to the end of this packet.<br>When this bit is set, it is expected that the packet's CRC would be supplied as part of the data downloaded to the FIFO. An exception to this occurs with a transmit underrun. In this case a guaranteed-bad CRC is appended to the packet.<br>When this bit is cleared, the NIC computes and appends CRCs for transmit packets. |

(1 of 2)

**3C90xB NICs FrameStartHeader Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [15] | txIndicate | When this bit is set, a txComplete interrupt occurs when a packet finishes transmitting. If this bit is cleared, no interrupt occurs unless a transmit error occurs. |
| [16] | dnComplete | This bit indicates that the packet download is complete. The NIC sets this bit after it has finished downloading all of the fragments specified in the DPD. |
| [25] | addIpChecksum | Setting this bit tells the NIC hardware to scan this packet and insert an IP checksum if it finds an IP header. |
| [26] | addTcpChecksum | Setting this bit tells the NIC hardware to scan this packet and insert a TCP checksum if it finds IP and TCP headers. |
| [27] | addUdpChecksum | Setting this bit tells the NIC hardware to scan this packet and insert a UDP checksum if it finds IP and UDP headers. |
| [28] | rndupDefeat | Setting this bit defeats the packet length round up operation. When this bit is cleared, transmit packet lengths are rounded up to an even number, according to the value of the rndupBndry bit.<br><br>For more information on packet length round up, see "Packet Length Round Up" later in this chapter. |
| [29] | dpdEmpty | This bit indicates that there is no packet data in this DPD entry, so the NIC should proceed directly to fetching the DnNextPtr DPD entry.<br><br>A driver can use this feature to allow it to handle the DownList-empty condition consistently. |
| [31] | dnIndicate | When this bit is set, a dnComplete interrupt occurs when a packet finishes downloading. If this bit is cleared, no interrupt occurs.<br><br>The NIC reads this bit after the download operation has finished, allowing the host to change this bit while the download is in progress.<br><br>The FSH is read twice: first to store the FSH bit values temporarily while the packet is being downloaded, and again after the download is finished to determine whether to generate an interrupt. |

(2 of 2)

## Schedule Time

The ScheduleTime DPD entry applies to 3C90xB NICs only.

This entry contains a field defining when the packet should be downloaded.

**ScheduleTime Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | | | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | |

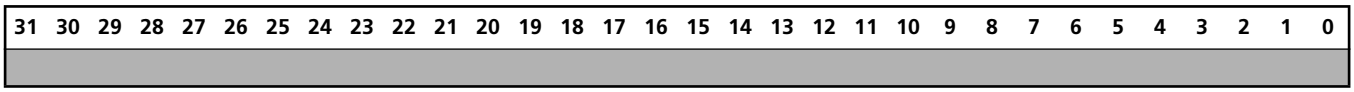
**ScheduleTime Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [23:0] | scheduleTime | This field provides either an absolute time at which to download this packet, or a value to be loaded into the RealTimeCnt register, depending upon the values of the loadTimeCnt and scheduleTimeValid bits.<br><br>When scheduleTimeValid is set, scheduleTime represents the time at which the packet is to be downloaded. When the NIC sees scheduleTimeValid set, it compares scheduleTime against the value in the RealTimeCnt register. If RealTimeCnt is greater than scheduleTime, the packet is downloaded. If RealTimeCnt is less, then the NIC goes into a polling mode, in which it periodically refetches the ScheduleTime entry and compares it against RealTimeCnt. The poll rate is determined by the value in the DnPoll register.<br><br>The scheduleTime bit specifies a time in increments of 800 ns. |
| [28] | loadTimeCnt | This bit, when set, instructs the NIC to load the value in the scheduleTime field into the RealTimeCnt register and download the packet immediately. When loadTimeCnt is set, scheduleTimeValid is ignored by the NIC. |
| [29] | scheduleTimeValid | This bit, when set, indicates that bits [23:0] contain a time schedule value.<br><br>When the NIC detects this bit set, it compares the value in scheduleTime with the current value in the RealTimeCnt register. If scheduleTime is less than RealTimeCnt, the NIC downloads the packet for transmission.<br><br>When this bit is clear, the NIC ignores the contents of scheduleTime, and the packet is downloaded immediately. |

## Down Fragment Address

The DnFragAddr DPD entry contains the physical address of a contiguous block of data to be downloaded to the NIC and transmitted.

**DnFragAddr Format**

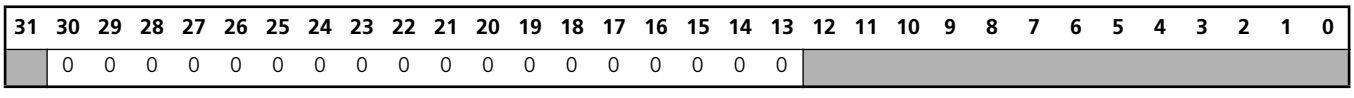



A fragment can start on any byte boundary.

## Down Fragment Length

The DnFragLen DPD entry contains fragment length and control information for the block of data pointed to by the previous DnFragAddr DPD entry.

**DnFragLen Format**



**DnFragLen Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [12:0] | dnFragLen | This field contains the length of the contiguous block of data pointed to by the previous DnFragAddr DPD entry. |
| [31] | dnFragLast | This bit is set by the driver to indicate that this is the last fragment of the transmit packet and that the NIC should proceed to the next DPD. |

## Packet Download

A packet download begins when all of the following conditions are true:

- The DnListPtr register is not equal to zero.
- The download engine is not in the DnStall command state.
- One of the following is true (3C90xB NICs only):
  - The DPD has no ScheduleTime entry.
  - There is a ScheduleTime DPD entry and the scheduleTime bit is less than the RealTimeCnt register value.
  - The loadTimeCnt bit is set in the ScheduleTime DPD entry.
- The transmit FIFO has more space available than the threshold specified in the DnBurstThresh register (3C90xB NICs only).

### Simple Packet Download

The simplest example of packet download starts with the download engine idle and an empty downlist, as would be the case after reset.

To download a single packet, the following actions occur:

1. The driver creates a DPD with the addresses and lengths of the buffers containing the data to be transmitted.

   Since there are no more DPDs, the driver programs zero into the DnNextPtr DPD entry.
2. The driver writes the address of the DPD into the DnListPtr register to start the download engine.
3. The NIC proceeds to fetch information from the DPD and move the packet data into the transmit FIFO.

### Packet Length Round Up

The NICs have the ability to round up the length of a transmit packet automatically. This is useful in some network operating system (NOS) environments in which packet lengths need to be made an even number.

#### 3C90x NICs

The 3C90x NIC drivers round up packet length to make the number of transmit bytes even.

If the sum of the fragment lengths is odd, drivers add one to it and write the result into the txLength field in the FrameStartHeader DPD entry. Because of the way earlier-generation NICs delimit packets in internal FIFO, they add one extra byte to the packet when the packet is transmitted on the network medium. The receiving station knows to strip this byte off the incoming packet because the true (odd) length of the packet is shown in the IEEE 802.3 Length field.

### 3C90xB NICs

The 3C90xB NIC drivers round up packet length in a way that is backward-compatible with the method used by 3C90x NICs.

The rounding up of the packet length is based on the sum of the fragment lengths specified in a DPD and the rndupBndry [1:0] field in the FrameStartHeader DPD entry. The rndupBndry bit coincides with the two low-order txLength bits written by earlier-generation NIC drivers. (The txLength field is not required by 3C90xB NICs.) Rounding up occurs when the packet length implied by the sum of the fragment lengths is odd and the value in rndupBndry is even. The packet length is rounded up to either a word or dword boundary, depending on the value of rndupBndry.

3C90xB NIC drivers may defeat rounding up of the length by setting the rndupDefeat bit in the FrameStartHeader DPD entry of each DPD.

## Download Scheduling

Transmit packets can be scheduled for download at a time determined by an on-chip real-time counter. A scheduled packet is not downloaded until the real-time counter is greater than or equal to the schedule time specified in the packet's DPD.

When a packet is at the head of the downlist waiting for its schedule time to be reached, all packets behind it are delayed. However, packets can be inserted in front of a scheduled packet to allow transmission of priority packets.

## Download Completion

After downloading a packet, the NIC writes a dnComplete bit into the DPD, eliminating the need for the driver to read the DnListPtr register to determine which DPDs have been downloaded.

The NIC can be configured to generate dnComplete interrupts when packets finish being downloaded. These dnComplete interrupts can be generated on a per-packet basis by programming the appropriate value into the dnIndicate field of each DPD's FrameStartHeader entry.

In response to a dnComplete interrupt, the driver acknowledges the interrupt and returns the DPD's buffers to the protocol. In the general case, in which the driver is using a multipacket downlist, when the driver enters its interrupt handler, multiple packets may have been downloaded. To determine which packets in a list of DPDs have been downloaded, the driver can traverse the list, examining the dnComplete bit in each DPD.

The NIC fetches the FrameStartHeader to examine the dnIndicate bit before packet download and again when the download is finished. This allows a driver to change dnIndicate while download of the packet is in progress. For example, a packet's DPD might be at the end of the downlist when it starts downloading, so the driver would probably set dnIndicate to generate an interrupt. However, if during the process of downloading this packet the driver were to add a new DPD to the end of the list, it might clear dnIndicate in the active DPD so that the interrupt is delayed until the next DPD finishes.

## Multipacket Lists

Generally, it is desirable for the driver to queue multiple DPDs. Multiple DPDs are linked together by pointing the DnNextPtr entry within each DPD at the next DPD, and programming zero into DnNextPtr in the last DPD.

Because the host and the NIC are generally both accessing the downlist at the same time, the host must stall the NIC before modifying the downlist or writing a new value to the DnListPtr register (unless the value is already zero). This is accomplished by issuing a DnStall command. When the host has finished manipulating the list, it issues a DnUnStall command.

The 3C90xB NICs add a new way to restart the download process when it has been idled by a zero DnNextPtr value. The NIC can be programmed to automatically poll on DnNextPtr until a nonzero value has been written to it. This polling function is controlled by the DnPoll register. The value written to DnPoll determines the DnNextPtr polling interval. The polling function is enabled whenever DnPoll contains a nonzero value. Enabling polling eliminates the need to stall the NIC.

### Adding DPDs to the End of the Downlist

You can add DPDs to the end of the downlist with polling disabled or enabled.

For 3C90x NICs, use the sequences for Polling Disabled.

***Polling Disabled*** The following sequence is recommended for adding DPDs to the downlist when download polling is disabled (the DnPoll register is zero):

1. Stall the download engine by issuing the DnStall command.
2. Wait for DnStall to finish by polling on the cmdInProgress bit in the IntStatus register.
3. Update the DnNextPtr entry in the last DPD in the downlist to point at the new DPD.
4. Read the DnListPtr register.
5. If DnListPtr is zero, write the address of the new DPD to DnListPtr.
6. Unstall the download engine by issuing the DnUnStall command.

   The download engine becomes idle when it fetches a zero value from a DnNextPtr DPD entry. One way to restart the download process is by writing a nonzero value to DnListPtr.

***Polling Enabled (3C90xB NICs only)*** When polling is enabled (the DnPoll register is nonzero), DPDs can be added with no register accesses. Point the last DPD's DnNextPtr entry at the new DPD.

### Inserting a DPD Near the Head of the Downlist

Although DPDs cannot be added at the head of the downlist, they can be added after the first active (unfinished) DPD.

***Polling Disabled*** The following sequence is recommended for inserting a DPD near the head of the downlist when download polling is disabled (the DnPoll register is zero):

1. Stall the download engine by issuing the DnStall command.
2. Wait for DnStall to finish by polling on the cmdInProgress bit in the IntStatus register.
3. Find the last DPD that is marked as downloaded (the dnComplete bit is one).
4. Update the DnNextPtr entry in this last DPD to point at the inserted DPD.
5. Point the inserted DPD's DnNextPtr entry where the last downloaded DPD points.
6. Read the DnListPtr register.
7. If DnListPtr is zero, write the address of the inserted DPD to DnListPtr.
8. Unstall the download engine by issuing the DnUnStall command.

***Polling Enabled (3C90xB NICs only)*** When polling is enabled (the DnPoll register is nonzero), DPDs can be inserted with no register accesses as follows:

1. Find the first DPD that is not marked as downloaded (the dnComplete bit is reset).
2. Set this DPD's DnNextPtr entry to zero.
3. Check to see if this DPD is now marked as downloaded. If so, it is too late to insert at this DPD. Restore DnNextPtr, and move to the next DPD in the list and restart this process. If the DPD is not downloaded, go to the next step.
4. Point the inserted DPD's DnNextPtr where the first DPD once pointed.
5. Point the first DPD's DnNextPtr at the inserted DPD, completing the chain.

#### Inserting a DPD in Front of a Scheduled DPD

The following sequence applies to 3C90xB NICs only.

A DPD can be inserted in front of a DPD that is scheduled to download next—that is, after a completed DPD and before a DPD whose download is being delayed until the value specified in the ScheduleTime DPD entry is reached.

1. Prepare the DPD to be inserted. Point its DnNextPtr entry at the scheduled DPD.
2. Set the DnNextPtr entry in the completed DPD to zero.
3. Read the DnListPtr register. If it no longer points to the completed DPD but instead points to the scheduled DPD, it is too late to insert at this location. Restore DnNextPtr in the completed DPD (if necessary, to keep the list coherent) and try inserting it after the next DPD.
4. If DnListPtr still points at the completed DPD, then complete the insertion by pointing DnNextPtr in the completed DPD at the DPD to be inserted.

#### Polling on DnNextPtr

The 3C90x NICs can restart the download process when it has been idled by a value of zero in the DnNextPtr entry.

The NICs can be programmed to automatically poll on DnNextPtr until a nonzero value has been written to it. This polling function is controlled by the DnPoll register. The value written to DnPoll determines the DnNextPtr polling interval. The polling function is enabled whenever DnPoll contains a nonzero value.

## NIC Download Sequence

The actions taken by the NIC hardware to download packets are described in this section.

### 3C90x NICs

Starting with the driver writing the DnListPtr register (for example, when starting from an empty downlist), the 3C90x NICs follow this sequence:

1. Check that the DnListPtr register is nonzero.
2. Check that it is not in the DnStall command state.
3. Check that the TxFree register value is greater than or equal to the value implied by the TxFreeThresh register.
4. Fetch the FrameStartHeader entry from the DPD pointed to by the DnListPtr register and write it to the transmit FIFO.
5. Fetch the DnFragAddr and DnFragLen entries one by one from the DPD, and move the associated data fragments to the transmit FIFO.
6. Issue an internal TxDone command and wait until the command is carried out.
7. If a DnStall command has been issued, wait until a DnUnStall command is issued.
8. If a transmit underrun has occurred, wait until the driver issues a TxReset command.
9. Fetch the FrameStartHeader again, and, if the dnIndicate bit is set, set the dnComplete indication (which may in turn cause an interrupt if the IndicationEnable and InterruptEnable masks are set correctly).
10. Fetch the DnNextPtr entry from the current DPD and load it into DnListPtr.
11. If the new DnListPtr is zero, the download engine becomes idle and waits for a nonzero value to be written to DnListPtr.
12. Repeat as necessary.

### 3C90xB NICs

Starting with the driver writing the DnListPtr register (for example, when starting from an empty downlist) the 3C90xB NICs follow this sequence:

1. Check that the DnListPtr register is nonzero.
2. Check that it is not in the DnStall command state.
3. Fetch the second dword from the DPD pointed to by the DnListPtr entry.

   If the dword contains FrameStartHeader information, it is written into the transmit FIFO. If the dword is a ScheduleTime entry, it checks the scheduleTimeValid and loadTimeCnt bits and takes the appropriate action, delaying the download if required. If the download is delayed, the download engine polls on the ScheduleTime DPD entry at a rate determined by the DnPoll register.
4. Fetch the DnFragAddr and DnFragLen DPD entries one by one from the DPD, and move the associated data fragments to the transmit FIFO.
5. Set the dnComplete bit in the DPD.
6. If a DnStall command has been issued, wait until a DnUnStall command is issued.
7. If a transmit underrun has occurred, wait until the driver issues a TxReset command.

8 Fetch the FrameStartHeader again, and, if the dnIndicate bit is set, set the dnComplete indication (which may in turn cause an interrupt if the IndicationEnable and InterruptEnable masks are set correctly).

9 Fetch the DnNextPtr entry from the current DPD.

If DnNextPtr is zero, the download engine becomes idle. If polling is disabled (the DnPoll register is zero), the download engine waits for a nonzero value to be written to the DnListPtr register. If polling is enabled (the DnPoll register is nonzero), the old value in DnListPtr is preserved, and the NIC polls on DnNextPtr in the DPD until it fetches a nonzero value from it.

If the value fetched from DnNextPtr is nonzero, then the value is stored temporarily in the NIC and the NIC inspects the DPD at that location. If the referenced DPD does not contain a ScheduleTime entry or it contains one that has already expired, then the temporary value is loaded into DnListPtr, advancing the NIC to the new DPD.

If the referenced DPD contains an unexpired ScheduleTime, then DnListPtr is not updated (the NIC stays at the old, completed DPD), and the NIC starts to time a polling interval (download polling must be enabled when ScheduleTime is used). When polling is complete, the NIC fetches DnNextPtr again (into the temporary register) and checks the referenced DPD again to see if its ScheduleTime has expired. This process is repeated until the ScheduleTime value is eventually reached. When this happens, the temporary value is finally loaded into DnListPtr, and the NIC advances to the new DPD.

10 With the new DPD to work on, the process begins again at step 2.

## Packet Transmission

The NIC initiates packet transmission (assuming transmission is enabled) as soon as either the entire packet is resident in the transmit FIFO, or the number of bytes that are resident is greater than the value in the TxStartThresh register.

### Enabling Transmission

The NIC comes out of reset with transmission disabled. Until transmission is enabled, no data is transmitted to the network medium. Any data downloaded to the NIC stays in the FIFO, waiting to be transmitted. If more data is downloaded than can fit into the FIFO, an overrun occurs.

Transmission is enabled with the TxEnable command (for 3C90x NICs, this occurs after the media type is set).

Transmission can be disabled with the TxDisable command. If TxDisable is issued while a packet transmission is in progress, it takes effect after the packet has been transmitted.

### Transmit Errors

When a transmit error occurs, a txComplete interrupt is generated, and the specific error is indicated by status bits in the TxStatus register.

To recover from a transmit error, the driver must reenable the transmitter and, in the case of an underrun or jabber error, reset the transmit logic with the TxReset command, before subsequent transmissions can occur.

With transmit errors that do not require TxReset (namely, the maxCollisions and txStatusOverflow bits in the TxStatus register), any pending packets in the transmit FIFO are preserved (except the packet that experienced maxCollisions), and they are transmitted after the transmitter is reenabled.

In general, download completions and transmit completions (including errors) are independent of one another—downloads operate on the tail end of the transmit FIFO, transmissions on the head. A special case is transmit underruns. When a transmit underrun occurs, that packet is the only one in the transmit FIFO, so the head and tail packets in the FIFO are the same.

### Underrun Recovery

If a transmit underrun error occurs, the NIC stops processing DPDs, and an interrupt is generated with a txUnderrun error flagged in the TxStatus register. By examining the current value of the DnListPtr register, the driver can determine which packet was being transmitted when the underrun occurred. All packet DPDs in the downlist before the underrun packet will have been downloaded successfully.

To recover from an underrun, the driver should follow this sequence:

1. Issue a DnStall command.
2. Ensure that the download and transmission processes are finished by polling on the dnInProg bit in the DmaCtrl register, and then polling on the txInProg bit in the MediaStatus register until they are cleared.
3. Issue a TxReset command to reset the underrun (this clears the dnComplete bit).
4. Reenable transmission by issuing a TxEnable command.
5. Restore all transmit-related thresholds (probably increasing the value in the TxStartThresh register, in particular).
6. Retransmit the packet by pointing the DnListPtr register at the DPD that experienced the underrun.

### Reclaiming Transmit FIFO Space

This applies to 3C90xB NICs only.

As a packet transmits out of the transmit FIFO, it is desirable to be able to release the packet data space so that it can be used for another packet download. However, if a collision occurs on a packet after part of it has been released, the NIC is unable to retransmit it and the packet must be downloaded again. This is inefficient.

The TxReclaimThresh register allows the driver to make trade-offs between efficiently using the transmit FIFO space and limiting the number of downloads due to collisions. The value programmed into TxReclaimThresh determines how much of a packet must be transmitted before its data starts to be released from the FIFO.

A txReclaimError occurs when a packet experiences a collision after its reclaim threshold has been crossed. For more information on reclaiming, see "TxReclaimThresh" in this chapter.

## Transmit Mechanism

The transmit mechanism allows the driver software to perform optimizations that reduce the number of interrupts generated.

### Limiting dnComplete Interrupts

The driver can limit the number of packets in the downlist for which a dnComplete interrupt is generated. It could, for example, only set the dnIndicate bit for the packet on the tail of the list (clearing dnIndicate for the current tail before enqueueing each new packet). Or the driver might require an interrupt every *n* packets. In any case, on each interrupt the driver would then dequeue all of the packets that were downloaded before that interrupt occurred (DPDs in which the dnComplete bit is set). Obviously there is a trade-off between latency and the number of interrupts taken—the driver writer is responsible for making this trade-off.

### Using CountDown Timer Instead of dnComplete

This applies to 3C90xB NICs only.

The driver can mask off dnComplete interrupts and use the CountDown register to generate interrupts instead. The driver might look at the time it would take to transmit all the bytes currently in the transmit FIFO and queued in the downlist and set CountDown to half that time, for example. The driver would then use the intRequested interrupt to dequeue all the DPDs in which the dnComplete bit is set. Again, this is just an example, and it is the driver writer's job to determine which algorithm to use.

## DmaCtrl

| | |
|---|---|
| Synopsis | Control and status register for bus master operations. (This register was PktStatus on earlier-generation NICs.) |
| Type | Read/write |
| Size | 32 bits |
| Offset | 20 |

DmaCtrl controls some of the functions in the upload and download engines, and contains some status bits.

DmaCtrl is cleared by a reset.

**DmaCtrl Register Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | 0 |

**DmaCtrl Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [1] | dnCmplReq | This read-only bit is set to the value that the packet controller reads from the dnIndicate field in the FrameStartHeader of the current DPD. |
| [2] | dnStalled | This read-only bit is set whenever downloading is stalled with the DnStall command. It is cleared by a DnUnstall command. |

**DmaCtrl Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [3] | upComplete | This read-only bit is the same as upComplete in the IntStatus register, except that this bit is always visible regardless of the setting of the IndicationEnable register.<br><br>This bit is different from the upPktComplete bit in the UpPktStatus register in that upComplete latches on once an upPktComplete indication has occurred.<br><br>This bit is cleared by issuing an AcknowledgeInterrupt command with the upCompleteAck bit set. |
| [4] | dnComplete | This read-only bit is the same as dnComplete in IntStatus, except that this bit is always visible regardless of the setting of the IndicationEnable register.<br><br>This bit is cleared by issuing an AcknowledgeInterrupt command with the dnCompleteAck bit set. |
| [5] | upRxEarlyEnable | This read/write bit determines when the NIC can start uploading a receive packet.<br><br>By default (cleared), uploads qualify for bus master arbitration when the packet becomes visible, normally at 60 bytes unless an RxEarlyThresh threshold smaller than that has been set.<br><br>When set to one, uploads do not start until the RxEarlyThresh threshold has been crossed (or the packet is completed, whichever is first). |
| [6] | armCountdown | This read-only bit specifies whether expiration of the Countdown register sets the intRequested bit.<br><br>If this bit is clear, Countdown expiration does not set the intRequested bit. If this bit is set, expiration of Countdown sets intRequested.<br><br>The armCountdown bit is completely managed by the hardware. This bit is cleared automatically by the act of setting intRequested, or when a zero value is written to Countdown. The armCountdown bit is set implicitly when a nonzero value is written to Countdown. |
| [7] | dnInProg | This read-only bit indicates that a download operation is in progress.<br><br>Drivers use this bit primarily in an underrun recovery routine. The driver waits for this bit to be cleared before issuing a TxReset command to clear the underrun condition.<br><br>Before checking this bit, issue a DnStall command to ensure that this bit is not set as a result of the NIC being in a polling mode. |
| [8] | counterSpeed | This read/write bit sets the count rate for the Countdown and FreeTimer registers.<br><br>When this bit is cleared, the count rate is once every 3.2 µs (four byte times at 10 Mbps). When counterSpeed is set, the count rate is once every 320 ns (four byte times at 100 Mbps).<br><br>By setting this bit appropriately for the negotiated wire speed, conversions can be made between byte times and counter values using simple shift operations. |

**DmaCtrl Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [9] | countdownMode | This read/write bit controls the operating mode of the Countdown register.<br><br>When this bit cleared, Countdown begins its down counting operation as soon as a nonzero value is written to it. When this bit set, Countdown does not begin counting down until the dnComplete bit in the IntStatus register is set.<br><br>For more information on the Countdown modes, see "Countdown" in Chapter 12. |
| [16] | upAltSeqDisable | This bit applies to 3C90xB NICs only.<br><br>Setting this bit disables the alternate upload sequence, so that the NIC mimics the behavior of 3C90x NICs.<br><br>When this bit is set, the upload engine writes the UpPktStatus UPD entry first and then fetches the UpNextPtr UPD entry. When this bit is clear (the default), the upload engine first fetches UpNextPtr and then writes UpPktStatus. |
| [20] | defeatMWI | This bit applies to 3C90xB NICs only.<br><br>Setting this read/write bit prevents the bus master logic from using the Memory Write Invalidate (MWI) PCI command. |
| [21] | defeatMRL | This bit applies to 3C90xB NICs only.<br><br>Setting this read/write bit prevents the bus master logic from using the Memory Read Line (MRL) PCI command. |
| [22] | upOverDiscDisable | This bit is for 3C90xB NICs only.<br><br>This read/write bit, when clear (the default), causes the upload engine to discard receive overrun packets without uploading them to memory. When this bit is set, the upload engine keeps and uploads overrun packets. |
| [30] | targetAbort | This read-only bit is set when the NIC experiences a target abort sequence when operating as a bus master. This bit indicates a fatal error, and it must be cleared before further download or upload operation can proceed.<br><br>This bit is cleared by issuing a GlobalReset command with the upDownReset mask bit cleared. |
| [31] | masterAbort | This read-only bit is set when the NIC experiences a master abort sequence when operating as a bus master. This bit indicates a fatal error, and it must be cleared before further download or upload operation can proceed.<br><br>This bit is cleared by issuing a GlobalReset command with the upDownReset mask bit cleared. |

(3 of 3)

## DnBurstThresh

| | |
|---|---|
| Synopsis | A threshold determining when bus master download requests are made. |
| Type | Read/write |
| Size | 8 bits |
| Offset | 2a |

**DnBurstThresh Register Format**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | | | | | |

The DnBurstThresh register applies to 3C90xB NICs only.

This register determines when the NIC makes download bus master requests, based upon the available space in the transmit FIFO. The value in this register represents free space in the FIFO in units of 32 bytes. When the free space exceeds the threshold, the NIC can make a download request.

DnBurstThresh may be overridden by the DnPriorityThresh register mechanism. See "PCI Bus Master Operation" in Chapter 3 for information about the relationship between DnBurstThresh and DnPriorityThresh.

A value of zero is invalid. DnBurstThresh defaults to 8, a threshold of 256 bytes.

## DnListPtr

| | |
|---|---|
| Synopsis | Points to the current DPD in the downlist. |
| Type | Read/write |
| Size | 32 bits |
| Offset | 24 |

**Type 0 DnListPtr Register Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 |

**Type 1 DnListPtr Register Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 | 0 |

The DnListPtr register holds the address of the current DPD in the downlist. The NIC interprets a value of zero in DnListPtr to mean that no more packets remain to be downloaded.

DnListPtr is cleared by reset.

Type 0 DnListPtr can only point to addresses on 8-byte boundaries, so DPDs must be aligned on 8-byte boundaries.

Type 1 DnListPtr can only point to addresses on 16-byte boundaries, so DPDs must be aligned on 16-byte boundaries.

DnListPtr may be written directly by host software to point the NIC at the head of a newly created downlist.

Writes to DnListPtr are ignored while the current value in the register is nonzero. To avoid access conflicts between the NIC and host software, the host must issue a DnStall command before writing to DnListPtr (unless the driver has specific knowledge that DnListPtr contains zero).

The NIC also updates DnListPtr while it processes DPDs in the downlist. As the NIC finishes processing a DPD, it fetches the value from the DnNextPtr entry. If the value is zero, the download engine becomes idle. Also, in 3C90xB NICs, if download polling is enabled (the DnPoll register is nonzero), the old value in DnListPtr is preserved.

If the value fetched from the DnNextPtr DPD entry is nonzero, then the value is stored temporarily in the NIC, and the NIC inspects the DPD at that location. If the referenced DPD does not contain a ScheduleTime entry or if it contains one that has already expired, then the temporary value is loaded into DnListPtr, and the NIC advances to the new DPD.

In 3C90xB NICs, if the referenced DPD contains an unexpired ScheduleTime entry, then DnListPtr is not updated (the NIC stays at the old, completed DPD), and the NIC starts to time a polling interval (any driver that uses ScheduleTime entries must also configure the NIC to poll). When the polling is complete, the NIC fetches DnNextPtr again (into the temporary register) and checks the referenced DPD again to see if the ScheduleTime value has expired. This process is repeated until the ScheduleTime value is eventually reached. When this happens, the temporary value is loaded into DnListPtr, and the NIC advances to the new DPD.

There are two ways the download engine can leave the idle state:

- The driver can write a nonzero value directly to DnListPtr.
- If polling is enabled, the download engine leaves the idle state when a nonzero value is finally fetched from DnNextPtr.

Reading DnListPtr while the download engine is polling for ScheduleTime expiration has the following side effects:

- Any pending decision to advance to the next DPD because the ScheduleTime value has just been reached is canceled.
- The download engine fetches the DnNextPtr entry in the current (completed) DPD immediately, rather than waiting for the full DnPoll interval. (It is assumed that the driver will only read DnListPtr when it is in the process of inserting a DPD at the list head, so it will have written a new value into DnNextPtr to hook up the inserted DPD).

*For Type 1, scheduled DPDs, the value written to DnListPtr must be 16-byte aligned.*

## DnMaxBurst

| | |
|---|---|
| Synopsis | Records the longest download (memory read) burst experienced by the NIC. |
| Type | Read/write |
| Size | 16 bits |
| Offset | 78 |

**DnMaxBurst Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | | | | | | | 0 | 0 | 0 | 0 | 0 |

The DnMaxBurst register applies to 3C90xB NICs only.

DnMaxBurst is a diagnostic register that records the longest memory read burst experienced by the NIC. The burst length is expressed in bytes, with a granularity of 32. DnMaxBurst may be cleared to restart the measurement process.

## DnPoll

| | |
|---|---|
| Synopsis | Sets the DnNextPtr DPD entry poll rate. |
| Type | Read/write |
| Size | 8 bits |
| Offset | 2d |

**DnPoll Register Format**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | | | | | | | |

The DnPoll register applies to 3C90xB NICs only.

The value in the DnPoll register determines the rate at which the current DPD is polled. DPDs are polled for two different reasons:

- When a zero DnNextPtr entry is fetched from the current DPD, DnNextPtr is polled to determine when a new DPD is ready to be processed.
- When packet download is delayed with the ScheduleTime DPD entry, the DPD is polled to determine when the RealTimeCnt has reached the required ScheduleTime value.

Polling is disabled when DnPoll is cleared. DnPoll is cleared by reset.

The value in DnPoll represents 320-ns time intervals. The maximum value represents 40.64 µs.

## DnPriorityThresh

| | |
|---|---|
| Synopsis | Provides a threshold to set a point at which the download engine makes a priority bus master request. |
| Type | Read/write |
| Size | 8 bits |
| Offset | 2c |

**DnPriorityThresh Register Format**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | | | | | | |

The DnPriorityThresh register applies to 3C90xB NICs only.

The value in the DnPriorityThresh register sets a point at which the download engine makes a priority bus master request. A priority download request has priority over the upload engine, unless the engine is also making a priority request. When the number of used bytes in the transmit FIFO falls below the value implied by DnPriorityThresh, the priority bus request is made.

A download priority request is not subject to the DnBurstThresh register constraint. When the FIFO is close to underrun, burst efficiency is sacrificed in favor of requesting the bus as quickly as possible.

The value in DnPriorityThresh represents data in the transmit FIFO in terms of 32-byte portions. DnPriorityThresh resets to 4, or a threshold of 128d bytes.

## TxFree

| | |
|---|---|
| Synopsis | Returns the space available in the transmit packet buffer area. |
| Type | Read-only |
| Size | 16 bits |
| Window | 3 |
| Offset | c (3C90x NICs also located at window 1, offset c) |

**TxFree Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | | | | | | | | |

The TxFree register provides a real-time indication of the number of bytes of free space that are available in the transmit FIFO. If this register returns zero, the transmit FIFO is full.

TxFree is unreliable while bus master operations are active.

## TxFreeThresh

| | |
|---|---|
| Synopsis | Controls when a packet download begins, based on how much free space exists in the transmit FIFO. |
| Type | Read/write |
| Size | 8 bits |
| Window | Extended register set |
| Offset | 2f |

**TxFreeThresh Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

The TXFreeThresh register applies to 3C90x NICs only.

The value in the TxFreeThresh register determines how much free space must exist in the transmit FIFO before the download engine can start a new packet download from the downlist.

The NIC scales the value in this register by 256 and compares the result with the value in the TxFree register. A packet download can begin when the value in TxFree is greater than or equal to the value implied by TxFreeThresh. For example, a TxFreeThresh value of one implies that there must be at least 256 bytes available in the transmit FIFO (and hence shown in TxFree) before a packet download begins.

Downloading is disabled entirely if TxFreeThresh implies a value greater than the transmit FIFO size.

The value in TxFreeThresh is examined only on packet boundaries and does not stop the next fragment in a packet from being downloaded.

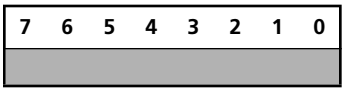
A value equivalent to approximately two packets works well in most cases. Drivers that do not need to insert new packets into the downlist might want to set just enough room for one maximum-sized packet to allow the FIFO to fill.

TxFreeThresh resets to all ones, which generally prevents any downloads from occurring. A driver should set TxFreeThresh to a reasonable value before using the download mechanism.

## TxPktId

| | |
|---|---|
| Synopsis | Allows read-back of the TxPktId field. |
| Type | Read-only |
| Size | 8 bits |
| Offset | 18 |

**TxPktId Register Format**




The TxPktId register applies to 3C90xB NICs only.

This register contains the packet ID for the currently transmitting or most recently transmitted packet. The TxPktId value comes from the pktId field in the packet's FrameStartHeader DPD entry.

Drivers can use TxPktId during transmit error recovery by scanning through the DPDs in the downlist, searching for a match between the TxPktId value and a pktId value.

A DPD may already be released by the time a collision is indicated. Therefore, to allow more efficient recovery from collisions, sequence numbers can be attached to transmit packets as follows:

- The driver writes a sequence number into the DPD.
- The NIC reads the sequence number, keeps it with the packet, and makes it available in the TxPktId register so that the driver can read it when a transmit error occurs.

## TxReclaimThresh

| | |
|---|---|
| Synopsis | Provides a threshold to control when transmit packet data is released from the FIFO. |
| Type | Read-only |
| Size | 8 bits |
| Window | 5 |
| Offset | 9 |

**TxReclaimThresh Register Format**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

The TxReclaimThresh register applies to 3C90xB NICs only.

The value in the TxReclaimThresh register determines how much of a packet must be transmitted before the data starts to be released for use by the tail of the FIFO.

TxReclaimThresh is set by issuing the SetTxReclaimThresh command.

The value in TxReclaimThresh represents a multiple of 16 bytes. A value of 255d in TxReclaimThresh disables the reclaim mechanism: packet space is not reclaimed until the entire packet is transmitted.

TxReclaimThresh resets to 8d, which yields a reclaim threshold of 128 bytes.

Once the number of bytes implied by the value in TxReclaimThresh has been transmitted, that number of bytes is discarded from the FIFO. Thereafter, bytes are discarded as they are transmitted to the network.

A txReclaimError (signaled in the TxStatus register) occurs when a packet experiences a collision after its reclaim threshold has been crossed, preventing it from being able to retry. When a reclaim error occurs, the transmitter is disabled, and the packet's ID number (sequence number) is visible in the TxPktId register. To recover from a reclaim error, the driver must issue a TxEnable command.

It is recommended that values no smaller than 4 be written to TxReclaimThresh, to avoid excessive reclaim errors due to in-window collisions.

## TxStartThresh

| | |
|---|---|
| Synopsis | Provides for an early transmission start based upon the number of packet bytes downloaded to the NIC. |
| Type | Read-only (write to advance queue) |
| Size | 16 bits |
| Window | 5 |
| Offset | 0 |

**TxStartThresh Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | | | | | | | | | | | | 0 | 0 |

The value in the TxStartThresh register is used to control early packet transmission. Transmission of a packet begins when the number of bytes for the packet downloaded to the NIC is greater than the value set in this register.

TxStartThresh is set using the SetTxStartThresh command.

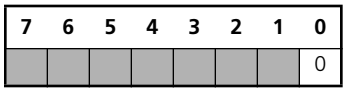
If TxStartThresh is set too low, the transmitter may experience underruns because the DMA data transfers are unable to keep up with the instantaneous wire data rate. Drivers should use underrun indications as a hint to increase the TxStartThresh value.

This register resets to 8188d, which disables the threshold mechanism.

## TxStatus

| | |
|---|---|
| Synopsis | Returns the transmit status for the current transmit packet. |
| Type | Read-only (write to advance queue) |
| Size | 8 bits |
| Offset | 1b |

**TxStatus Register Format**




The TxStatus register returns the status of packet transmission attempts. TxStatus actually implements a queue of up to 31 transmit status bytes. An I/O write of an arbitrary value to TxStatus advances the queue to the next transmit status byte.

**TxStatus Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [1] | txReclaimError | This bit applies to 3C90xB NICs only.<br>This bit indicates that a transmit reclaim error occurred, meaning that the packet experienced a collision after the front of the packet had already been reclaimed to the FIFO free space. |
| [2] | txStatusOverflow | This bit, when set, indicates that the TxStatus stack is full, and as a result, the transmitter has been disabled. Writing the TxStatus register clears this bit, but the transmitter must be reenabled with the TxEnable command before transmissions can resume. |
| [3] | maxCollisions | This bit, when set, indicates that a packet was not successfully transmitted, because it encountered 16 collisions. The TxEnable command must be issued to recover from this condition. The packet is discarded from the transmit FIFO, so typically software should resubmit the packet for transmission. |
| [4] | txUnderrun | This bit indicates that the packet experienced an underrun during the transmit process—the host was unable to supply the packet data fast enough to keep up with the network. An underrun halts the transmitter and the transmit FIFO. The TxReset and TxEnable commands must be issued before any new packets are submitted to the NIC. |

(1 of 2)

**TxStatus Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [5] | txJabber | This bit is asserted if the NIC determines that it is transmitting for too long. The TxReset command is required to recover from this error. |
| [6] | interruptRequested | This bit is asserted if the txIndicate bit was set when the 32-bit FrameStartHeader was written to the NIC for the packet in question. |
| [7] | txComplete | If this bit is false, then the remainder of the status bits are undefined. If the host chooses to poll this register while waiting for a packet transmission to finish, then this bit is used to determine whether a packet transmission attempt has finished that either experienced an error or had the txIndicate bit set in the transmit packet descriptor. |

(2 of 2)

# 7 RECEPTION AND UPLOAD

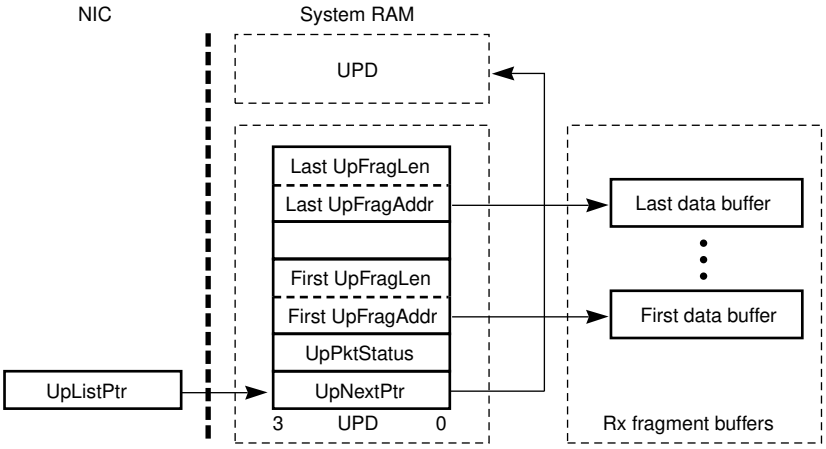
This chapter presents an overview of the packet reception process and defines the registers associated with the reception and uploading of data.

The NICs support a multipacket multifragment scatter process, whereby incoming packets are moved to system memory buffers defined by descriptors. The descriptors themselves also reside in system memory, and are linked together by the host CPU.

## Packet Upload Model

The packet upload mechanism is similar to the download mechanism. Upload is structured around a linked list of packet descriptors, called upload packet descriptors (UPDs). UPDs contain pointers to the fragment buffers into which the NIC is to place receive data. The linked list of UPDs, called the uplist, is illustrated in Figure 28.

**Figure 28** Uplist



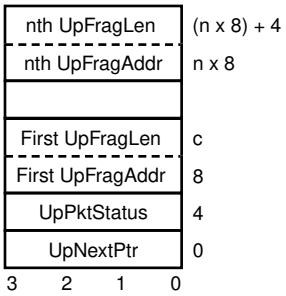

The head of the uplist is the UPD that corresponds to the current upload packet. The Up List Pointer (UpListPtr) register points to this UPD. As the UPD is processed, the fragment address and fragment length values are fetched one by one from the UPD into on-NIC registers, which are used to control the data upload operations.

When the NIC exits reset, the upload engine is in the idle state, ready to start processing an uplist as soon as a nonzero value is written into UpListPtr.

## UPD Data Structure

A UPD is 16 to 512 bytes long. It contains the UpNextPtr and UpPktStatus entries, and from 1 to 63 pairs of UpFragAddr and UpFragLen entries. See Figure 29.

**Figure 29** UPD Format




### Up Next Pointer

The first dword in the UPD is the UpNextPtr entry, which contains the physical address of the next UPD in the uplist. If this is the last UPD in the uplist, then this value is zero.

**UpNextPtr Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 |

UPDs must be aligned on 8-byte physical address boundaries.

### Up Pkt Status

The second dword in the UPD is the UpPktStatus entry. At the end of a packet upload, the NIC writes the value of the UpPktStatus register into this location in the UPD.

**UpPktStatus Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 0 | 0 | | | | | | | | 0 | | | | | | | | | | | | | |

**UpPktStatus Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [12:0] | upPktLen | This field is the number of packet bytes uploaded.<br>This is essentially the packet length, except when the packet is larger than the number of bytes specified in the upload fragments. In this case, the upOverflow bit is set. |
| [14] | upError | This bit indicates that an error occurred in the receipt of the packet. The driver should examine bits [16:20] of this register to determine the specific errors. |
| [15] | upComplete | This bit indicates that the packet upload is complete. |

(1 of 3)

**UpPktStatus Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [16] | upOverrun | This bit indicates that the hardware was unable to remove data from the receive FIFO quickly enough, resulting in lost data. Bytes are missing from the packet at one or more (unpredictable) locations in the packet.<br><br>When this bit is set, it is most likely because the software failed to provide a UPD quickly enough, or it kept the NIC in the UpStall state for too long. |
| [17] | runtFrame | This bit indicates that the packet was a runt (less than 60 bytes). Normally such frames are not uploaded unless the RxEarlyThresh register is set to less than 60. |
| [18] | alignmentError | This bit indicates that the packet had an alignment error (a bad CRC plus dribble bits). |
| [19] | crcError | This bit indicates a CRC error on the packet. |
| [20] | oversizedFrame | This bit indicates that the packet was larger than the maximum allowable size, as defined in the MaxPktSize register. |
| [23] | dribbleBits | This bit indicates that the packet had accompanying dribble bits. This bit is informational only and does not indicate a packet error. |
| [24] | upOverflow | This bit indicates that the UPD specified insufficient buffer space for the packet—there were still bytes left to be uploaded when the NIC ran out of buffers. When this bit is set, the NIC has discarded the remainder of the packet. |
| [25] | ipChecksumError | This bit applies to 3C90xB NICs only.<br><br>This bit indicates that the packet contained an error in the IP header checksum. This bit is only valid when the ipChecksumChecked bit is set. |
| [26] | tcpChecksumError | This bit applies to 3C90xB NICs only.<br><br>This bit indicates that the packet contained an error in the TCP header checksum. This bit is only valid when the tcpChecksumChecked bit is set. |
| [27] | udpChecksumError | This bit applies to 3C90xB NICs only.<br><br>This bit indicates that the packet contained an error in the UDP header checksum. This bit is only valid when the udpChecksumChecked bit is set. |

**UpPktStatus Bit Descriptions (continued)**

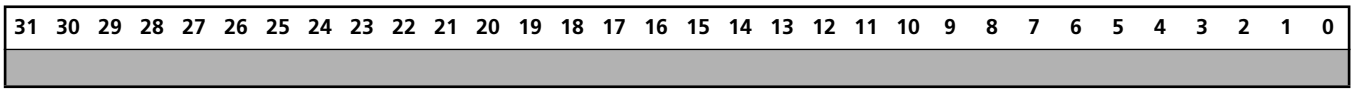
| Bit | Name | Description |
|---|---|---|
| [28] | impliedBufferEnable | This bit applies to 3C90xB NICs only.<br>This bit enables a special upload mode that reduces the number of information fetches by the NIC, and is intended for server applications in which packets are received into a ring of maximum-packet-sized buffers.<br>Setting this bit instructs the NIC not to fetch any UpFragAddr or UpFragLen entries from this UPD. Instead, the NIC assumes that there is one receive buffer of length 1528d bytes, starting immediately after the UpPktStatus entry at (UPD address + 8).<br>The driver sets this bit when it prepares the UPD. The NIC tests this bit before uploading a packet. At the same time, the NIC also tests the upComplete bit.<br>When the NIC updates the UpPktStatus entry at the end of the upload operation (in order to set the upComplete bit), the value written to this bit is undefined. A driver cannot assume a certain value is left in this bit after the UPD is used. Therefore, the driver must write the desired value to this bit every time it releases a UPD to the NIC. |
| [29] | ipChecksumChecked | This bit applies to 3C90xB NICs only.<br>This bit, when set, indicates that the packet contained an IP header. The ipChecksumError bit in the UpPktStatus register contains the result of the checksum comparison. |
| [30] | tcpChecksumChecked | This bit applies to 3C90xB NICs only.<br>This bit, when set, indicates that the packet contained a TCP header recognizable by the NIC (fragmented TCP datagrams do not set this bit). The tcpChecksumError bit in the UpPktStatus register contains the result of the checksum comparison. |
| [31] | udpChecksumChecked | This bit applies to 3C90xB NICs only.<br>This bit, when set, indicates that the packet contained a UDP header recognizable by the NIC (fragmented UDP datagrams do not set this bit). The udpChecksumError bit in the UpPktStatus register contains the result of the checksum comparison. |

(3 of 3)

### Up Fragment Address

The third (fifth, and so on) dword in the UPD, UpFragAddr, contains the physical address of a contiguous block of system memory to which receive data is to be uploaded.

**UpFragAddr Format**




A fragment can start on any byte boundary.

### Up Fragment Length

The fourth (sixth, and so on) dword in the UPD, UpFragLen, contains fragment length and control information for the block of data pointed to by the previous UpFragAddr UPD entry.

**UpFragLen Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | |

**UpFragLen Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [12:0] | upFragLen | This field contains the length of the contiguous block of data pointed to by the previous UpFragAddr UPD entry. |
| [31] | upLastFrag | This bit is set by the driver to indicate that this is the last fragment of the receive packet. |

## Packet Reception

The following sections describe various aspects of packet reception.

### Enabling Reception

The 3C90x NICs come out of reset with reception and the StationMask register disabled.

The 3C90xB NICs come out of reset with reception disabled.

Until reception is enabled, no incoming packets are accepted by the NIC. Once reception (and the StationMask register for 3C90x NICs) is enabled, packets are received according to the value programmed in the RxFilter register.

Reception is enabled by issuing the RxEnable command and can be disabled with the RxDisable command. If RxDisable is issued while a packet is being received, the disabling takes effect after the reception has finished.

### Simple Packet Upload

The simplest example of packet upload starts with the upload engine idle and an empty uplist, as would be the case after a reset.

To upload a single packet, the following actions occur:

1. The driver creates a UPD with the addresses and lengths of the buffers to be used (typically one buffer, equal to the maximum packet size).

   Because there are no more UPDs, the driver programs zero into the UpNextPtr UPD entry.
2. The driver writes the address of the UPD into the UpListPtr register.
3. Assuming that there is a receive packet in the FIFO, the NIC proceeds to fetch information from the UPD and move the packet data into the buffers.

With receives, it is likely that the driver needs to set up one or more UPDs and their associated buffers before reception of a packet. One approach is to simply allocate a block of full-size packet buffers in its own data space and create UPDs that point to the buffers. Another approach is for the driver to request the buffers from the protocol ahead of time.

Similar to download, there are UpStall and UpUnStall commands. The driver should issue an UpStall command before modifying the list pointers in the uplist.

As with download, the upload engine becomes idle if it fetches an UpListPtr register of zero.

**Upload Eligibility**

The upRxEarlyEnable bit in the DmaCtrl register controls when a packet upload can begin. By default, upRxEarlyEnable is clear. With this setting, an upload can begin when the packet becomes visible. Normally, this is when 60 bytes have been received, unless the RxEarlyThresh register is set to a smaller value.

When upRxEarlyEnable is set, uploads do not start until RxEarlyThresh has been crossed (or the packet completes reception, whichever is first).

In either case, setting RxEarlyThresh to a value less than 60 may cause the host to process an excessive number of collision fragments.

**Packet Upload Completion**

The NIC can be configured to generate an upComplete interrupt when a packet upload is completed. In response to an upComplete interrupt, the driver looks at the UpPktStatus UPD entry to determine the size of the packet and whether there were any errors, and then copies the packet out of the buffers, if needed.

In general, when the driver enters its interrupt handler, multiple packets may have been uploaded. The driver can read the UpListPtr register to determine which UPDs in the list have been used. The driver starts at the head of its UPD list and traverses backward until it reaches the UPD whose address matches the UpListPtr register. However, because I/O operations are costly, it is more efficient to use the upComplete bit in each UPD to determine which packets have been uploaded.

**Multipacket Lists**

Generally, it is desirable for the driver to create a list of multiple UPDs. Multiple UPDs are linked together by pointing the UpNextPtr entry within each UPD at the next UPD and programming zero into UpNextPtr in the last UPD.

One upload option that differs from download is that the uplist can be formed into a ring. The NIC does an implicit UpStall command if it starts to process a UPD that has already been used (one in which the upComplete bit is set in the UpPktStatus UPD entry). Or, if the new UpPoll register is set to a nonzero value, the NIC does not stall but automatically rechecks upComplete periodically until it is cleared.

When the driver finishes processing a UPD, it should leave the UpPktStatus entry cleared, and if the UpPoll register is zero, it should issue an UpUnStall command, just in case the NIC has already read the UpPktStatus UPD entry and stalled.

The following sequence is recommended for adding UPDs to the uplist:

1. Stall the upload engine by issuing the UpStall command.
2. Update the UpNextPtr entry in the last UPD in the uplist to point at the new UPD.
3. Read the UpListPtr register.
4. If UpListPtr was zero, write the address of the new UPD into UpListPtr.
5. Unstall the upload engine by issuing the UpUnStall command.

Most drivers probably simply allocate a number of full-size packet buffers, create a UPD for each one, and link the UPDs into a ring. As packets are received and uploaded, an upComplete interrupt is generated for each one.

**Early Receive Interrupts**

The NICs can be programmed to generate an interrupt based upon the number of bytes that have been received in a packet. The RxEarlyThresh register sets this early receive threshold.

**Parallel Tasking of Receive Uploads**

Some drivers need to be able to copy a packet out of the scatter buffer into the protocol buffer while the packet is still being uploaded (an example is the DOS ODI client driver). The UpPktStatus register is provided for this purpose.

If the driver issues an UpStall command, reads the UpListPtr and UpPktStatus registers, and then issues an UpUnStall command (in other words, reads UpListPtr and UpPktStatus in a critical section), the driver can determine how much of the packet has been uploaded. If UpListPtr is pointing to a UPD for an incomplete packet, then UpPktStatus gives the number of bytes uploaded so far. (If not, the packet has been completely updated, and UpPktStatus in the UPD should be examined instead.) The driver can then do memory copies out of the buffer in parallel with the upload operation.

**NIC Upload Sequence**

The NIC performs the following steps to upload a packet to the host:

1. Checks that the UpListPtr register is nonzero.
2. Checks that the NIC is not in the UpStall state.
3. Resets the UpPktStatus register.
4. Fetches UpPktStatus from the current UPD.

   If the upComplete bit is set, then the NIC does an implicit UpStall, stalling the upload process. If the UpPoll register contains a nonzero value, the NIC then polls on the upComplete bit and waits for it to be cleared before continuing. Otherwise, the upload process continues.
5. Waits for the top receive packet to become eligible for upload (for details, see "Upload Eligibility" earlier in this chapter).
6. Uploads the packet into the fragments specified in the UPD, or into the implied buffer if the impliedBufferEnable bit in the UpPktStatus register is set. If there is more data in the packet than space in the fragment buffers, the NIC generates an upOverflow error.
7. As the packet is being uploaded, maintains the UpPktStatus register, specifically the upPktLen field.
8. At the end of packet upload, updates the UpPktStatus register with any error code from the packet and sets the upComplete bit.
9. Issues an internal RxDiscard command and waits for it to finish.
10. If the upAltSeqDisable bit in the DmaCtrl register is set, writes UpPktStatus to the UPD in host memory. (This behavior is compatible with 3C90x NICs.)
11. If an UpStall command has been carried out, waits until an UpUnStall command has been executed.

12 Fetches the UpNextPtr entry from the UPD. If UpNextPtr is zero and polling is enabled, the NIC starts a polling loop. If polling is disabled, loads the fetched value into the UpListPtr UPD entry.

13 If the upAltSeqDisable bit in the DmaCtrl register is clear, writes UpPktStatus to the UPD in host memory. (This behavior is compatible with 3C90xB NICs.)

14 If a polling loop had been started, polls on UpNextPtr until a nonzero value is fetched, and loads the value into UpListPtr.

15 If the UpListPtr value is zero (polling is disabled), then the upload engine becomes idle and waits for a nonzero value to be written into UpListPtr.

16 Repeats as necessary.

## DmaCtrl

See "DmaCtrl" in Chapter 6.

## MaxPktSize

| | |
|---|---|
| Synopsis | Determines the receive packet size that will flag the upOversizedPkt bit. |
| Type | Read/write |
| Size | 16 bits |
| Window | 3 |
| Offset | 4 |

**MaxPktSize Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | | | | | | | | | | | | | |

The MaxPktSize register applies to 3C90xB NICs only.

The value in MaxPktSize determines the minimum receive packet size that is flagged as oversize.

MaxPktSize defaults to 1514d upon reset. For backward compatibility with earlier-generation NICs, MaxPktSize is automatically loaded with 1514d or 4491d upon the RxReset command, depending on the value of the allowLargePackets bit in the MacControl register.

## RxEarlyThresh

| | |
|---|---|
| Synopsis | Returns the value of the RxEarlyThresh register. |
| Type | Read-only |
| Size | 16 bits |
| Window | 5 |
| Offset | 6 |

**RxEarlyThresh Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | | | | | | | | | | | | 0 | 0 |

The value stored in this register defines the number of bytes that must be received before an rxEarly indication occurs. The first byte of the destination address is considered to be byte 1.

RxEarlyThresh can be set using the SetRxEarlyThresh command.

RxEarlyThresh resets to the value 8188d, which disables the threshold mechanism.

As soon as the number of bytes that have been received is greater than the value in RxEarlyThresh, the NIC generates an rxEarly interrupt to the host (assuming the rxEarly indication and interrupt bits are not masked). The rxEarly interrupt occurs only when the packet being received is the top packet; in other words, only if the packet being received can be transferred by the host during reception.

The RxEarlyThresh mechanism causes one rxEarly indication per packet unless it is retriggered. The rxEarly interrupt is meant to be used as a retriggerable interrupt. In other words, it is legal for the driver to respond to an rxEarly interrupt resulting from a value set in the RxEarlyThresh register, and then reprogram RxEarlyThresh to a larger value so that a subsequent interrupt is generated within the same receive packet. If a new value is set in RxEarlyThresh while a packet is being received from the medium, then an rxEarly interrupt is generated as soon as the rxEarly threshold is crossed, or immediately if the threshold was already crossed.

An rxEarly indication occurs whenever the RxEarlyThresh threshold has been met and the packet being received is the top packet.

The driver can program any value into RxEarlyThresh, but setting RxEarlyThresh to less than 8 causes the NIC to interpret the value as 8, to allow the NIC to perform destination address filtering before generating an rxEarly indication.

On 3C90xB NICs, the value in RxEarlyThresh may also determine how many bytes of a packet must be received before upload transfers for the packet are allowed to begin. If the upRxEarlyEnable bit in the DmaCtrl register is set, a packet is not eligible to start upload until the number of bytes defined in the UpPktStatus DPD entry has been received.

Setting RxEarlyThresh to a value that is too low causes the host to respond to the interrupt before the entire receive packet header has been received. Setting RxEarlyThresh to a value that is too high introduces unnecessary delays in the system's receive response sequence.

If RxEarlyThresh is set to a value that is greater than the length of the received packet, then an rxComplete interrupt (rather than an rxEarly interrupt) occurs at the completion of packet reception.

If the host system is particularly slow in responding to an rxEarly interrupt, then it is likely that the packet has been completely received by the time the driver examines the NIC. In this case, rxEarly is overridden by rxComplete. The rxEarly and rxComplete interrupts are mutually exclusive. Because rxEarly goes away when rxComplete becomes set, rxComplete should only be disabled if rxEarly is also disabled. Such disabling prevents spurious interrupts.

## RxError

| | |
|---|---|
| Synopsis | Returns error and informational bits for the top receive frame. |
| Type | Read-only |
| Size | 8 bits |
| Window | 1 |
| Offset | 4 |

**RxError Register Format**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | 0 | 0 | | | | | |

The RxError register applies to 3C90x NICs only.

**RxError Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | rxOverrun | This bit indicates that software was unable to remove data from the receive FIFO quickly enough, so a data loss condition resulted. |
| [1] | runtFrame | This bit indicates that the frame was less than 60 bytes. |
| [2] | alignmentError | This bit indicates that the frame had an alignment error. |
| [3] | crcError | This bit indicates a CRC error on the frame. |
| [4] | oversizedFrame | This bit indicates that the frame was larger than the maximum allowed size.<br>Table 21 gives the minimum frame size at which an error is flagged. The frame size includes the destination and source address and type/length field, but does not include the FCS field. |
| [7] | dribbleBits | This bit indicates that the frame had accompanying dribble bits. This bit is informational only; it does not indicate a frame error. |

**Table 21** Minimum Frame Size for RxError

| allowLargePackets Bit Value (from MacControl Register) | Minimum oversizedFrame Size |
|---|---|
| 0 | 1515 |
| 1 | 4491* |

* This value was calculated by subtracting bytes for fields that have no Ethernet equivalent from the maximum FDDI frame size of 4500.

## RxFilter

| | |
|---|---|
| Synopsis | Defines the types of receive packets that are accepted. |
| Type | Read-only |
| Size | 8 bits |
| Window | 5 |
| Offset | 8 |

**RxFilter Register Format**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | | | | | |

**RxFilter Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | receiveIndividual | Setting this bit enables the NIC to receive packets that match the station address set for the NIC. |
| [1] | receiveMulticast | Setting this bit causes the NIC to receive all multicast packets, including broadcast. |
| [2] | receiveBroadcast | Setting this bit causes the NIC to receive all broadcast packets. |
| [3] | receiveAllFrames | Setting this bit causes the NIC to receive all packets in promiscuous mode. |
| [4] | receiveMulticastHash | This bit applies to 3C90xB NICs only.<br>Setting this bit enables the NIC to receive packets that pass the hash filtering mechanism. |

Each bit in the RxFilter register, when set, enables reception of a different type of packet.

RxFilter is set using the SetRxFilter command. It is cleared upon reset.

## RxFree

| | |
|---|---|
| Synopsis | Returns the space available in the receive packet buffer area. |
| Type | Read-only |
| Size | 16 bits |
| Window | 3 |
| Offset | a |

**RxFree Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | | | | | | | | |

The RxFree register provides a real-time indication of the number of bytes of free space that are available in the receive FIFO. If zero is returned, the receive FIFO is full.

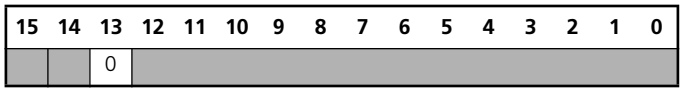
RxFree must be read as a 16-bit quantity to guarantee a valid return value.

*On 3C90x NICs, RxFree is unreliable while bus master operations are active.*

## RxStatus

| | |
|---|---|
| Synopsis | Provides frame status information. |
| Type | Read-only |
| Size | 16 bits |
| Window | 1 |
| Offset | 8 |

**RxStatus Register Format**




The RxStatus register applies to 3C90x NICs only.

The RxStatus register returns the status and the number of bytes in the FIFO for the top receive frame. This register may be read multiple times throughout the process of transferring the frame from the NIC to the host. RxStatus must be read as a 16-bit quantity to ensure the reliable transfer of the dynamic frame length information from the NIC to the host.

**RxFilter Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [12:0] | rxBytes | This field returns the number of data bytes that are available in the receive FIFO for the top frame.<br><br>Because it is possible to read data from the FIFO when the NIC is still receiving the same frame, it is likely that after the NIC has read a block of data from the FIFO, the rxBytes field indicates that a greater number of bytes remains to be read rather than a smaller number.<br><br>As the top frame is being received from the medium, bits [1:0] of this field are unreliable and the driver should mask them. When the frame reception is finished, this field reflects the exact number of bytes remaining in the FIFO.<br><br>Frames are padded to dword boundaries with the receive FIFO. It is legal for the driver to read the pad bytes when it is reading frame data from the FIFO. This improves driver efficiency in some cases. It is illegal to read beyond the pad bytes. |
| [14] | rxError | This bit indicates that an error occurred on the top receive frame, as indicated in the RxError register. This bit is the logical OR of bits [4:0] in the RxError register. |
| [15] | rxIncomplete | This bit indicates that the frame available in the FIFO and defined in the RxStatus register is currently being received from the network. When this bit is asserted, the rxError bit is zero. If this bit is false, rxError becomes valid. |

## StationAddress

| | |
|---|---|
| Synopsis | Defines the NIC's station address for receive purposes. |
| Type | Read/write |
| Size | 48 bits (accessible as three words) |
| Window | 2 |
| Offset | 0, 2, 4 |

The StationAddress register is used to define the individual destination address that the NIC responds to when receiving packets. Network addresses are generally specified in the form 00:20:af:12:34:56, where the bytes are received left to right, and the bits within each byte are received right to left (least-significant bit to most-significant bit).

StationAddress is written with three separate word accesses. To use the address above as an example, a driver would perform the following writes to StationAddress:

- A write of 2000h to offset 0
- A write of 12afh to offset 2
- A write of 5634h to offset 4

The writes can be made in any order; the important consideration is that the individual bytes end up in the correct byte position within the register.

The value programmed into StationAddress is *not* inserted into the source address field of packets transmitted by the NIC. The NIC's source address must be specified for every packet as part of the packet contents.

*On 3C90x NICs, the StationAddress register must be accessed with no larger than word-wide cycles.*

## StationMask

| | |
|---|---|
| Synopsis | Defines a mask to apply to the station address register. |
| Type | Read/write |
| Size | 48 bits (accessible as 3 words) |
| Window | 2 |
| Offset | 6, 8, a |

The StationMask register allows bits in receive packets to be treated as "don't cares" during individual address matching. Setting a bit in StationMask causes the value in the corresponding bit of StationAddress to be ignored when the destination address of incoming packets is compared with the NIC's individual address.

StationMask is written in the same way as the StationAddress register, using three separate word accesses to offsets 6, 8 and a.

In 3C90x NICs, the value must be set explicitly. In 3C90xB NICs, the default is 0.

## UpBurstThresh

| | |
|---|---|
| Synopsis | A threshold determining when bus master upload requests are made. |
| Type | Read/write |
| Size | 8 bits |
| Offset | 3e |

**UpBurstThresh Register Format**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | | | | | |

The UpBurstThresh register applies to 3C90xB NICs only.

This register determines when the NIC makes upload bus master requests, based upon the number of used bytes in the receive FIFO. The value in UpBurstThresh represents used space in the FIFO in units of 32 bytes. When the used space exceeds the threshold, the NIC may make an upload request on the PCI bus.

UpBurstThresh may be overridden by the UpPriorityThresh register mechanism. For information about the relationship between UpBurstThresh and UpPriorityThresh, see "PCI Bus Master Operation" in Chapter 3.

A value of zero is invalid. UpBurstThresh defaults to 8, a threshold of 256 bytes.

## UpListPtr

| | |
|---|---|
| Synopsis | Points to the current UPD in the uplist. |
| Type | Read/write |
| Size | 32 bits |
| Offset | 38 |

**UpListPtr Register Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 0 | 0 | 0 |

The UpListPtr register holds the physical address of the current UPD in the uplist. A value of zero in UpListPtr is interpreted by the NIC to mean that no more UPDs are available to accept receive packets.

UpListPtr is cleared by reset.

UpListPtr can only point to addresses on 8-byte boundaries, so UPDs must be aligned on 8-byte physical address boundaries.

UpListPtr may be written directly by host software to point the NIC at the head of a newly created uplist.

UpListPtr is also updated by the NIC as it processes UPDs in the uplist. As the NIC finishes processing a UPD, it loads UpListPtr with the value from the UpNxtPtr UPD entry to allow it to move on to the next UPD. If the NIC loads a value of zero from the current UPD, the upload engine enters the idle state, waiting for a nonzero value to be written to UpListPtr.

To avoid access conflicts between the NIC and host software, the host must issue an UpStall command before writing to UpListPtr.

## UpMaxBurst

| | |
|---|---|
| Synopsis | Measures the longest upload (memory write) burst on the PCI bus. |
| Type | Read/write |
| Size | 16 bits |
| Offset | 7a |

**UpMaxBurst Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | | | | | | | 0 | 0 | 0 | 0 | 0 |

The UpMaxBurst register applies to 3C90xB NICs only.

This register records the longest memory write burst experienced by the NIC. The burst length is expressed in bytes, with a granularity of 32. UpMaxBurst may be cleared to restart the measurement process.

## UpPktStatus

| | |
|---|---|
| Synopsis | Indicates the status of upload operations. |
| Type | Read-only |
| Size | 32 bits |
| Offset | 30 |

**UpPktStatus Register Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 0 | | | | | | 0 | 0 | | | | | | | | | | | | | | | | | | | | | |

**UpPktStatus Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [12:0] | upPktLen | This field gives a real-time indication of the number of bytes uploaded for the packet during packet upload.<br><br>This bit is cleared when the NIC fetches a new UpListPtr register, and counts up in steps no larger than a bus master burst. When the packet has been completely uploaded, this bit indicates the true packet length. |
| [13] | upStalled | This bit is asserted whenever the NIC is in the UpStall state, either because of an UpStall command or because of an implicit stall due to fetching a UPD with the upPktComplete bit in the UpPktStatus register already set.<br><br>This bit is cleared with an UpUnStall command. |

**UpPktStatus Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [14] | upError | This bit indicates that an error occurred in the receipt of the packet. The driver should examine bits [16:20] of this register to determine the type of errors. |
| [15] | upPktComplete | This bit indicates that the packet is complete. Unless an upload stall is in effect, this bit normally remains on only momentarily (too short for the software to read it) because the hardware then fetches the next UPD. |
| [16] | upOverrun | This bit indicates that the hardware was unable to remove data from the receive FIFO quickly enough (most likely because the software failed to free a UPD quickly enough, or kept the NIC in the UpStall state for too long). Bytes are missing from the packet at one or more locations in the packet (unpredictable). |
| [17] | upRuntFrame | This bit indicates that the packet was a runt (less than 60 bytes). Normally such frames are not uploaded unless RxEarlyThresh is set to a value less than 60. |
| [18] | upAlignmentError | This bit indicates that the packet had an alignment error (a bad CRC plus dribble bits). |
| [19] | upCRCError | This bit indicates a CRC error on the packet. |
| [20] | upOversizedFrame | ■ On 3C90x NICs, this bit indicates that the packet was larger than the maximum allowable size.<br>Normally packets are flagged as oversized at 1515 bytes or longer (not including the FSH). However, if the allowLargePackets bit is set, then packets are flagged as oversized at 4491 bytes or longer.<br>■ On 3C90xB NICs, this bit indicates that the packet was equal to or greater than the value set in the MaxPktSize register. |
| [23] | dribbleBits | This bit indicates that the packet had accompanying dribble bits. This bit is informational only, and does not indicate a packet error. |
| [24] | upOverflow | This bit indicates that the UPD had insufficient buffer storage for the packet—there were still bytes left to be uploaded when the NIC ran out of fragments. The NIC uploads what it can into the buffers provided, discards the rest, and sets this bit. |
| [25] | ipChecksumError | This bit applies to 3C90xB NICs only.<br>This bit indicates that the packet contained an error in the IP header checksum. This bit is only valid when ipChecksumChecked is set. |
| [26] | tcpChecksumError | This bit applies to 3C90xB NICs only.<br>This bit indicates that the packet contained an error in the TCP header checksum. This bit is only valid when tcpChecksumChecked is set. |
| [27] | udpChecksumError | This bit applies to 3C90xB NICs only.<br>This bit indicates that the packet contained an error in the UDP header checksum. This bit is only valid when udpChecksumChecked is set. |
| [29] | ipChecksumChecked | This bit applies to 3C90xB NICs only.<br>This bit, when set, indicates that the packet contained an IP header, and ipChecksumError contains the result of the checksum comparison. |

**UpPktStatus Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [30] | tcpChecksumChecked | This bit applies to 3C90xB NICs only.<br>This bit, when set, indicates that the packet contained a TCP header recognizable by the NIC (fragmented TCP datagrams do not set this bit), and tcpChecksumError contains the result of the checksum comparison. |
| [31] | udpChecksumChecked | This bit applies to 3C90xB NICs only.<br>This bit, when set, indicates that the packet contained a UDP header recognizable by the NIC (fragmented UDP datagrams do not set this bit), and udpChecksumError contains the result of the checksum comparison. |

(3 of 3)

The error bits (14 and [20:16]) are undefined until the upPktComplete bit is set.

Bits [27:25] and [31:29] are undefined until the upPktComplete bit is set.

UpPktStatus shows the status of various logic in the upload logic. Drivers should read this register only while the upload engine is in the UpStall state. Otherwise, the hardware may change UPDs between accesses to this register. The format of this register is identical to that of the UpPktStatus field written into processed UPDs, except that the impliedBufferEnable bit is not implemented here.

UpPktStatus is cleared by a reset.

## UpPoll

| | |
|---|---|
| Synopsis | Allows setting of the upComplete poll rate. |
| Type | Read/write |
| Size | 8 bits |
| Offset | 3d |

**UpPoll Register Format**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | | | | | | | |

The UpPoll register applies to 3C90xB NICs only.

The value in the UpPoll register determines the rate at which the current UPD is polled when the NIC is looking for the upComplete bit in the UpPktStatus register to be cleared.

Polling is disabled when UpPoll is cleared. UpPoll is cleared by reset.

The value in UpPoll represents 320-ns time intervals. The maximum representable value is 40.64 µs.

## UpPriorityThresh

| | |
|---|---|
| Synopsis | Provides a threshold to control when the upload engine makes a priority bus master request. |
| Type | Read/write |
| Size | 8 bits |
| Offset | 3c |

**UpPriorityThresh Register Format**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | | | | | |

The UpPriorityThresh register applies to 3C90xB NICs only.

The value in the UpPriorityThresh register sets a point at which the upload engine makes a priority bus master request. A priority upload request has priority over all other requests on the NIC. The priority bus request is made when the free space in the receive FIFO falls below the value in UpPriorityThresh.

An upload priority request is not subject to the UpBurstThresh constraint: when the FIFO is close to overrun, burst efficiency is sacrificed in favor of requesting the bus as quickly as possible.

The value in UpPriorityThresh represents free space in the receive FIFO in terms of 32-byte portions. UpPriorityThresh resets to 4, or a threshold of 128d bytes.

## VlanMask

| | |
|---|---|
| Synopsis | Provides the ability to mask reception of individual VLAN IDs in the 3Com proprietary VLAN Tagging (VLT) scheme. |
| Type | Read/write |
| Size | 16 bits |
| Window | 7 |
| Offset | 0 |

The VlanMask register applies to 3C90xB NICs only.

Under the 3Com proprietary VLAN Tagging (VLT) scheme, each packet includes a 4-bit VLAN ID field. The NIC's default behavior, when VLT is enabled, is to receive all VLT packets, regardless of the contents of the ID field. VlanMask allows individual IDs to be masked, which causes the NIC to discard packets containing those ID values.

VlanMask is cleared upon reset. When the vltEnable bit in the MacControl register is clear, VlanMask is ignored.

Each bit in VlanMask corresponds to a VLT ID value. Setting a bit causes packets containing the corresponding ID value to be discarded. Bit 0 corresponds to an ID value of zero, bit 1 corresponds to an ID value of one, and so on.

# 8 INTERRUPTS AND INDICATIONS

This chapter provides an overview of interrupts and indications, and defines the registers associated with interrupts.

Indications are reports of any interesting events on the NIC. An indication appears as a set bit in the IntStatus register. Indications can be individually masked off to prevent them from appearing as set in IntStatus. For 3C90xB NICs, there are eight different types of indications.

Any indication can be individually configured to cause an interrupt, which is the actual assertion of the interrupt signal on the PCI bus.

In this technical reference, the term *interrupt* is used loosely to refer to both interrupts and indications. It is assumed that a driver configures the NIC to generate an interrupt for any indication that is of interest to it.

When responding to an interrupt, the host reads the IntStatus register to determine the cause of the interrupt. In the IntStatus register, there are eight bits that define the source of the interrupt. The least-significant bit, interruptLatch, is always set whenever any of the interrupts are asserted. This is done to prevent spurious interrupts on the host bus. The interruptLatch bit must be explicitly acknowledged (cleared) using the AcknowledgeInterrupt command.

The host acknowledges interrupts by carrying out the interrupt-specific actions summarized in Table 22.

**Table 22** Interrupt-specific Actions

| Action | Description |
|---|---|
| interruptLatch | Acknowledged by the AcknowledgeInterrupt command |
| hostError | Acknowledged by issuing the appropriate resets |
| txComplete | Acknowledged by writing to the TxStatus register |
| rxComplete | Acknowledged automatically by the hardware |
| rxEarly | Acknowledged by the AcknowledgeInterrupt command |
| intRequested | Acknowledged by the AcknowledgeInterrupt command |
| updateStats | Acknowledged by reading one or more statistics registers |
| linkEvent | Acknowledged by reading the AutoNegExpansion register. This action applies to 3C90xB NICs only. |
| dnComplete | Acknowledged by the AcknowledgeInterrupt command |
| upComplete | Acknowledged by the AcknowledgeInterrupt command |

A two-level enable structure gives drivers flexibility in configuring indications and interrupts. For example, the driver may want one type of indication to interrupt the host processor, a second indication to not cause an interrupt but still be visible when the IntStatus register is read, and a third indication to be completely ignored.

## IndicationEnable

| | |
|---|---|
| Synopsis | Specifies which bits in the IntStatus register can become set. |
| Type | Read-only |
| Size | 16 bits |
| Window | 5 |
| Offset | c |

**IndicationEnable Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | 0 |

The IndicationEnable register allows the eight indication bits to be individually masked off so that they appear as zero in the IntStatus register, even though the corresponding condition in the IntStatus register is true. In order for an indication bit to be set in IntStatus, its corresponding bit-position in IndicationEnable must be set.

The IndicationEnable register is written by issuing the SetIndicationEnable command.

Each bit set in IndicationEnable enables the corresponding bit to be set in the IntStatus register. This register is set using the SetIndicationEnable command. See "SetIndicationEnable" in Chapter 10 for more details.

IndicationEnable is cleared upon reset.

## InterruptEnable

| | |
|---|---|
| Synopsis | Specifies which bits in the IntStatus register can generate an interrupt to the host. |
| Type | Read-only |
| Size | 16 bits |
| Window | 5 |
| Offset | a |

**InterruptEnable Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | 0 |

The InterruptEnable register controls which of the eight indication bits (after passing through the IndicationEnable register) can generate an interrupt. In order for an indication bit to generate an interrupt, its corresponding bit-position in the InterruptEnable register must be set.

The InterruptEnable register is written by issuing the SetInterruptEnable command.

Each bit in InterruptEnable is the interrupt enable bit for the corresponding bit in the IntStatus register. Setting a bit in InterruptEnable allows that source to generate an interrupt on the bus. This register is set using the SetInterruptEnable command. See "SetInterruptEnable" in Chapter 10 for more details.

InterruptEnable is cleared upon reset. It is also cleared by a read of the IntStatusAuto register.

## IntStatus

| | |
|---|---|
| Synopsis | Indicates the sources for NIC interrupts, and the number of the visible register window. |
| Type | Read-only |
| Size | 16 bits |
| Window | All |
| Offset | e |

**IntStatus Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 0 | | | | | | | | 0 | | | |

**IntStatus Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | interruptLatch | This bit is set when the NIC is driving the bus interrupt signal. It is a logical OR of the interrupt-causing bits after they have been filtered through the InterruptEnable register.<br>This bit is acknowledged by issuing the AcknowledgeInterrupt command with the interruptLatchAck bit set. |
| [1] | hostError | This bit is set when a catastrophic error related to the bus interface occurs.<br>The errors that set this bit are PCI target abort and PCI master abort.<br>This bit is cleared by issuing the GlobalReset command with the upDownReset mask bit cleared. |
| [2] | txComplete | This bit is set when a packet (whose txIndicate bit in the FrameStartHeader is set) has been successfully transmitted or for any packet that experiences a transmission error.<br>This interrupt is acknowledged by writing to the TxStatus register to advance the status FIFO. |
| [4] | rxComplete | This bit is set when one or more entire packets have been received into the receive FIFO.<br>This bit is automatically acknowledged by the upload engine as it uploads packets. Drivers should disable this interrupt and mask this bit when reading IntStatus. |

(1 of 3)

**IntStatus Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [5] | rxEarly | This bit is set when the number of bytes of the top packet that have been received is greater than the value of the RxEarlyThresh register.<br><br>When the top packet has been completely received by the NIC, this bit is negated and the rxComplete bit asserts (assuming that the appropriate masks are clear).<br><br>This bit is acknowledged by issuing the AcknowledgeInterrupt command with the rxEarlyAck bit set. |
| [6] | intRequested | This bit is set by the execution of a RequestInterrupt command or by the expiry of the Countdown register.<br><br>This bit is acknowledged by issuing the AcknowledgeInterrupt command with the intRequestedAck bit set. |
| [7] | updateStats | This bit indicates that one or more of the statistics counters is nearing an overflow condition (typically half of its maximum value). Reading all of the statistics acknowledges this bit.<br><br>A driver should respond to an updateStats interrupt by reading all of the statistics. This has the side effect of acknowledging (clearing) updateStats. |
| [8] | linkEvent | This bit applies to 3C90xB NICs with the 40-0502-00x ASIC only.<br><br>This bit indicates a change in the link status, as detected by the on-chip auto-negotiation logic.<br><br>A change in link status is defined as either entering the LINK GOOD state (meaning that auto-negotiation has been completed), or leaving the LINK GOOD state, because of a link failure.<br><br>Drivers should determine the cause of a linkEvent interrupt by checking the two interrupt status bits in the AutoNegExpansion register. Reading AutoNegExpansion automatically clears linkEvent.<br><br>Note that the auto-negotiation logic can generate linkEvent even when the xcvrSelect bit in the InternalConfig register is not set for auto-negotiation. It is the driver's responsibility to mask off linkEvent interrupts when it is not interested in receiving them.<br><br>This bit is not implemented in 3C90x and 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC. It always returns a zero. |
| [9] | dnComplete | This bit indicates that a packet download has been completed, and the DPD in question has had the dnIndicate bit set in its FrameStartHeader.<br><br>This bit is acknowledged by an AcknowledgeInterrupt command with the dnComplete bit set.<br><br>The host should examine the DnListPtr register to determine which packets have been downloaded—those in the downlist before the current DnListPtr (which if zero, implies all those in the list) have already been downloaded. |

**IntStatus Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [10] | upComplete | This bit indicates that a packet upload has been completed. This bit is acknowledged by an AcknowledgeInterrupt command with the upComplete bit set. |
| [12] | cmdInProgress | This bit indicates that the last command issued is still being executed by the NIC.<br>This bit need only be checked after one of the commands that require longer than a single I/O cycle to finish has been issued. No new commands can be issued until this bit is negated. |
| [15:13] | windowNumber | This field indicates which set of registers is currently visible in the I/O space of the NIC.<br>These bits are reset after a hardware reset or a GlobalReset command. |

(3 of 3)

IntStatus is the main status register for the NIC. It indicates the source of interrupts and indications on the NIC, the completion status of commands issued to the Command register, and the current register window visible in the lower part of the I/O space.

Bits [1:10] are the interrupt-causing sources for the NIC. These bits can be individually disabled as interrupt sources using the InterruptEnable register, and individually forced to read as zero in IntStatus using the IndicationEnable register.

IntStatus is cleared by reset.

## IntStatusAuto

| | |
|---|---|
| Synopsis | Special version of the IntStatus register with some added side effects to allow a reduction in the number of I/O operations required to service interrupts. |
| Type | Read-only |
| Size | 16 bits |
| Offset | 1e |

This register applies to 3C90xB NICs only.

The IntStatusAuto register actively acknowledges the active interrupts in the IntStatus register and clears the InteruptEnable register.

IntStatusAuto has the same bit definition as IntStatus. It differs from IntStatus only in the following side effects that occur when it is read:

- The InterruptEnable register is cleared. This prevents subsequent events from generating an interrupt on the bus.
- The following bits in IntStatus (if they are set) are acknowledged (cleared): dnComplete, upComplete, rxEarly, intRequested, and interruptLatch.

# 9 STATISTICS AND DIAGNOSTICS

This chapter provides an overview of statistics and defines the registers associated with statistics and diagnostics.

The NIC includes statistic counters of various widths. The gathering of statistics is enabled by issuing the StatisticsEnable command. When enabled, the statistic counters advance as corresponding events occur. No host intervention is required to facilitate this counting.

Reading a statistic register clears the register. Writing a value to a statistic register adds that value to the register. This is useful in diagnostics and IC production tests. Reading all of the statistics acknowledges the updateStats interrupt.

It is not necessary to disable statistics collection while reading the statistic registers. It is legal to do so, but disabling statistics collection may result in missed statistics events.

Whenever one or more of the statistic registers reaches half of its maximum value, an updateStats interrupt is generated.

Transmit and receive statistics are summarized in Table 23 and Table 24 and are described in alphabetical order in this chapter.

**Table 23** Summary of Transmit Statistics

| Statistic | Description |
|---|---|
| BytesXmittedOk | A byte total for all packets transmitted without error. |
| CarrierLost | A count of packets that were transmitted without error but experienced a loss of carrier. |
| FramesDeferred | A count of events where the transmission of a packet had to defer to network traffic. A single packet may defer more than once as a result of collisions, and each deference would be counted. |
| FramesXmittedOk | The number of packets of all types transmitted without errors. Loss of carrier and absence of an expected SQE are not considered errors. |
| LateCollisions | A count of every occurrence of a late collision (there could be more that one per packet transmitted). |
| MultipleCollisions | A count of all packets transmitted without error after experiencing from 2 through 15 collisions (including late collisions). |
| SingleCollisions | A count of packets that are transmitted without errors after one and only one collision (including late collisions). |
| SqeErrors | A count of events that occur if the NIC is configured to expect an SQE pulse after each transmission and did not receive such a pulse. |
| UpperBytesOk | A display of the high-order bits of the BytesRcvdOk and BytesXmittedOk statistics. |
| UpperFramesOk | A display of the high-order bits of the FramesRcvdOk and FramesXmittedOk statistics. |

**Table 24** Summary of Receive Statistics

| Statistic | Description |
|---|---|
| BadSSD | A count of packets received with bad start-of-stream delimiter. This statistic is only valid for 100BASE-TX or 100BASE-FX operation. |
| BytesRcvdOk | A byte total for all packets received without error. A packet's bytes are included in this count if the packet is received without errors. |
| FramesRcvdOk | A count of packets of all types that are received without error. |
| RxOverruns | A count of rxOverrun errors. Only packets that the host actually sees as overruns are included in this count, and not packets that are completely ignored by the NIC because the receive FIFO is full. |

## BadSSD

| | |
|---|---|
| Synopsis | Indicates the number of packets that experienced an error in the start-of-stream delimiter. |
| Type | Read/write |
| Size | 8 bits |
| Window | 4 |
| Offset | c |

This statistic register counts the number of packets that are received with a bad start-of-stream delimiter. This statistic is only valid when the NIC is operating in 100BASE-TX or 100BASE-FX mode.

This is an 8-bit counter and wraps around to zero after reaching ffx. An updateStats interrupt occurs after the counter has counted through 80h.

Reading this statistic clears it. Therefore, this statistic must be read as an 8-bit quantity. The StatisticsEnable command must have been issued for this register to count events.

*This register is not implemented in 3C900B NICs.*

## BytesRcvdOk

| | |
|---|---|
| Synopsis | Indicates the total number of bytes for frames that are received without error. |
| Type | Read/write |
| Size | 16 bits |
| Window | 6 |
| Offset | a |

This statistic register counts the number of bytes that are received successfully. For the purposes of this statistic, a successfully received packet is one that is completely moved into the receive FIFO before being discarded by the hardware.

This is a 16-bit counter and wraps around to zero after reaching ffffh. An updateStats interrupt occurs after the counter has counted through 8000h.

Reading this statistic clears it. Therefore, this statistic must be read as a 16-bit quantity. The StatisticsEnable command must have been issued for this register to count events.

## BytesXmittedOk

| | |
|---|---|
| Synopsis | Indicates the total number of bytes for packets that are transmitted without error. |
| Type | Read/write |
| Size | 16 bits |
| Window | 6 |
| Offset | c |

This statistic register counts the number of bytes included in packets that are transmitted with no errors reported in the TxStatus register.

This is a 16-bit counter and wraps around to zero after reaching ffffh. An updateStatistics interrupt occurs after the counter has counted through 8000h.

Reading this statistic clears it. Therefore, this statistic must be read as a 16-bit quantity. The StatisticsEnable command must have been issued for this register to count events.

## CarrierLost

| | |
|---|---|
| Synopsis | Indicates the number of packets experiencing loss of carrier during transmission. |
| Type | Read/write |
| Size | 8 bits |
| Window | 6 |
| Offset | 0 |

**CarrierLost Register Format**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | |

This statistic register counts the number of packets that experience at least one loss of carrier during transmission.

Carrier sense is not monitored for the purpose of this statistic until after the preamble and start-of-frame delimiter. This 4-bit counter sticks at 0fh. An updateStatistics indication occurs after the counter has counted through 08h.

Reading this statistic clears it. The StatisticsEnable command must have been issued for this register to count events.

## FramesDeferred

| | |
|---|---|
| Synopsis | The number of transmit packets deferred to network activity. |
| Type | Read/write |
| Size | 8 bits |
| Window | 6 |
| Offset | 8 |

This statistic register counts the number of times a transmit packet must defer to network traffic. A single packet may cause multiple deferrals as a result of collisions and retransmissions.

This is an 8-bit counter and wraps around to zero after reaching ffh. An updateStatistics interrupt occurs after the counter has counted through 80h. Reading this statistic clears it.

The StatisticsEnabled command must have been issued for this register to count events.

## FramesRcvdOk

| | |
|---|---|
| Synopsis | The number of error-free packets received. |
| Type | Read/write |
| Size | 8 bits |
| Window | 6 |
| Offset | 7 |

This statistic register counts the number of packets that are received without error. Packets received with errors are defined as packets in which one of the following bits is set:

- For 3C90x NICs (in the RxError register): rxOverrun, runtFrame, alignmentError, crcError, or oversizedFrame
- For 3C90xB NICs ( in the UpPktStatus register): upOverrun, upRuntFrame, upAlignmentError, upCRCError, or upOversizedFrame

This 10-bit counter wraps around to zero after reaching 3ffh. An updateStatistics indication occurs after the counter counts through 200h.

The low-order eight bits of this register are visible at this location. The upper two bits are visible in the UpperFramesOk register.

When FramesRcvdOk is read, the value in the upper two bits of that register is latched and made visible in UpperFramesOk. This latched value can be read from UpperFramesOk at any time until FramesRcvdOk is again read.

Reading UpperFramesOk has no effect on the value seen in UpperFramesOk. See "UpperFramesOk" in this chapter for more information.

The StatisticsEnable command must have been issued for this register to count events.

## FramesXmittedOk

| | |
|---|---|
| Synopsis | The number of error-free packets transmitted. |
| Type | Read/write |
| Size | 8 bits |
| Window | 6 |
| Offset | 6 |

This statistic register counts the number of packets that are transmitted without error. Error packets are defined as those for which the maxCollisions, txJabber, or txUnderrun bit is set to one in the TxStatus register.

This is a 10-bit counter that wraps around to zero after reaching 3ffh. An updateStatistics indication occurs after the counter counts through 200h.

The low-order eight bits of this register are visible within the register. The upper two bits are visible in the UpperFramesOk register.

When the FramesXmittedOk register is read, the value in the upper two bits of the register is latched and made visible in UpperFramesOk. This latched value can be read from UpperFramesOk at any time until FramesXmittedOk is again read.

Reading UpperFramesOk has no effect on the value seen in UpperFramesOk. See "UpperFramesOk" in this chapter for more information.

The StatisticsEnable command must have been issued for FramesXmittedOk to count events.

## LateCollisions

| | |
|---|---|
| Synopsis | Returns the number of late collisions during transmission attempts. |
| Type | Read/write |
| Size | 8 bits |
| Window | 6 |
| Offset | 4 |

This statistic register counts the number of late collisions. Since every transmission attempt is monitored, it is possible to count multiple late collisions per transmit packet.

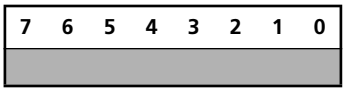
This 8-bit counter wraps around to zero after reaching ffh. An updateStatistics indication occurs after the counter counts through 80h.

Reading this statistic clears it. The StatisticsEnabled command must have been issued for this register to count events.

## MultipleCollisions

| | |
|---|---|
| Synopsis | The number of transmit packets experiencing at least two collisions. |
| Type | Read/write |
| Size | 8 bits |
| Window | 6 |
| Offset | 2 |

**MultipleCollisions Register Format**




This statistic register counts the number of packets that are transmitted successfully after experiencing anywhere from 2 through 15 collisions or late collisions.

This 8-bit counter wraps around to zero after reaching ffh. An updateStatistics indication occurs when the counter has counted through 80h. Reading this statistic has the side effect of clearing it.

The StatisticsEnable command must have been issued for this counter to be enabled.

## RxOverruns

| | |
|---|---|
| Synopsis | Counts the number of packets that cause an rxOverrun error. |
| Type | Read/write |
| Size | 8 bits |
| Window | 6 |
| Offset | 5 |

This statistic register counts the number of packets that should have been received (the destination address matched the filter criteria) but experienced an rxOverrun error because there was not enough FIFO space to hold the packet.

This statistic only includes overruns that become apparent to the driver, and does not count packets that are completely ignored because the receive FIFO is full at the start of packet reception.

This 8-bit counter wraps around to zero after reaching ffh. An updateStatistics indication occurs after the counter has counted through 80h.

Reading this statistic clears it. The StatisticsEnable command must have been issued for this register to count events.

## SingleCollisions

| | |
|---|---|
| Synopsis | Returns the number of packets experiencing a single collision. |
| Type | Read/write |
| Size | 8 bits |
| Window | 6 |
| Offset | 3 |

**SingleCollisions Register Format**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

This statistic register counts the number of packets that are transmitted without error after experiencing a single collision.

This 8-bit counter wraps around to zero after reaching ffh. An updateStatistics interrupt occurs after the counter has counted through 80h.

Reading this statistic clears it. The StatisticsEnable command must have been issued for this register to count events.

## SqeErrors

| | |
|---|---|
| Synopsis | Counts the number of transmit packets that experience SQE errors. |
| Type | Read/write |
| Size | 8 bits |
| Window | 6 |
| Offset | 1 |

**SqeErrors Register Format**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | |

This statistic register counts the number of transmit packets that result in an SQE error.

This is a 4-bit counter that sticks at 0fh. An updateStatistics interrupt occurs after the counter has counted through 08h.

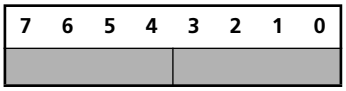
Reading this statistic clears it. The StatisticsEnable command must have been issued for this register to count events.

SqeErrors collection can be disabled independently of other statistics by clearing the enableSqeStats bit in the MediaStatus register. Normally, SqeErrors would only be enabled if an external transceiver over the AUI was used.

## UpperBytesOk

| | |
|---|---|
| Synopsis | Makes visible the high-order bits of the BytesRcvdOk and BytesXmittedOk statistic registers. |
| Type | Read-only |
| Size | 8 bits |
| Window | 4 |
| Offset | d |

**UpperBytesOk Register Format**




**UpperBytesOk Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [3:0] | upperBytesRcvdOk | The high-order four bits of the BytesRcvdOk register. This value is latched whenever BytesRcvdOk is read. |
| [7:4] | upperBytesXmittedOk | The high-order four bits of the BytesXmittedOk register. This value is latched whenever BytesXmittedOk is read. |

This statistic register allows read access to the high-order bits of the BytesRcvdOk and BytesXmittedOk statistics.

UpperBytesOk is cleared by reset.

The BytesRcvdOk and BytesXmittedOk registers are actually 20-bit registers. Their lower 16 bits are visible in the BytesRcvdOk and BytesXmittedOk registers in window 6.

See the "BytesRcvdOk" and "BytesXmittedOk" register definitions for more details on how UpperBytesOK works.

## UpperFramesOk

| | |
|---|---|
| Synopsis | Makes visible the high-order bits of the FramesRcvdOk and FramesXmittedOk statistic registers. |
| Type | Read-only |
| Size | 8 bits |
| Window | 6 |
| Offset | 9 |

**UpperFramesOk Register Format**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | | | 0 | 0 | | |

**UpperFramesOk Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [1:0] | upperFramesRcvdOk | The high-order two bits of the FramesRcvdOk register. This value is latched whenever FramesRcvdOk is read. |
| [5:4] | upperFramesXmittedOk | The high-order two bits of the FramesXmittedOk register. This value is latched whenever FramesXmittedOk is read. |

This statistic register allows read access to the high-order bits of the FramesRcvdOk and FramesXmittedOk statistic registers.

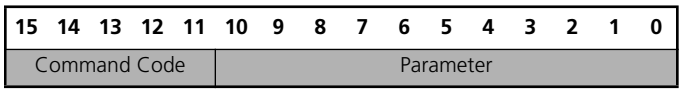
See the "FramesRcvdOk" and "FramesXmittedOk" register definitions for details about how the register values are latched into UpperFramesOk.

# 10 COMMAND REGISTER

This chapter provides an overview of the Command register and gives definitions of the commands.

| | |
|---|---|
| Synopsis | Allows commands to be issued to the NIC. |
| Type | Write-only |
| Size | 16 bits |
| Window | All |
| Offset | e |

Format




The Command register is used to issue commands of various types to the NIC. Commands may or may not contain parameters. Most commands execute in less time than it takes for the host system to perform a subsequent read or write operation and are considered to execute in zero time. Those commands that take a nonzero amount of time to execute are identified in their respective definitions.

All commands must be issued as a single write to the Command register. If the command being issued has Xs occupying bits [7:0], then a write to only bits [15:8] (offset fh) can be used. If any of the least-significant eight bits of the command word are defined, then a single 16-bit write must be used. The read-only IntStatus register is located with the Command register.

The command definitions in this chapter use the following conventions:

- The bit value is the 16-bit value that the NIC expects to be written to the Command register to carry out the desired operation. The most-significant five bits make up the Command Code; the remaining bits are the parameter.
- Bit positions occupied by an "X" indicate that the value for the corresponding bit does not matter. However, for future hardware compatibility, it is recommended that zeros be written to these positions.
- Bit positions occupied by a dot (•) indicate bit positions that are to be filled by the parameter associated with the command.

The commands are summarized in Table 25 and described in alphabetical order in the following sections.

*Commands in Table 25 that are marked with an asterisk (*) (for example, GlobalReset *) are not always completely executed before the next command can be issued to the NIC. For these commands, the driver must ensure that the cmdInProgress bit in the IntStatus register is a zero before taking any further action with the NIC.*

**Table 25** Command Summary

| Command Type | Command Name | Bit Value | Description |
|---|---|---|---|
| Reset | GlobalReset * | (0000 000• •0•• ••••) * | Perform an overall reset of NIC. |
| | RxReset * | (0010 100• 0000 ••••) * | Reset the receive logic. |
| | TxReset * | (0101 100• 0000 ••••) * | Reset the transmit logic. |
| Transmit | DnStall * | (0011 0XXX XXX0 0010)* | Stall the download engine. |
| | DnUnStall | (0011 0XXX XXX0 0011) | Unstall the download engine. |
| | SetTxReclaimThresh | (1100 0000 •••• ••••) | Set the value of the TxReclaimThresh register. This command applies to 3C90xB NICs only. |
| | SetTxStartThresh | (1001 1••• •••• ••••) | Set the value of the TxStartThresh register. |
| | TxDisable | (0101 0XXX XXXX XXXX) | Disable packet transmission. |
| | TxEnable | (0100 1XXX XXXX XXXX) | Enable packet transmission. |
| Receive | RxDisable | (0001 1XXX XXXX XXXX) | Disable packet reception. |
| | RxEnable | (0010 0XXX XXXX XXXX) | Enable packet reception. |
| | SetHashFilterBit | (1100 1•XX XX•• ••••) | Program a particular bit in the hash filter. This command applies to 3C90xB NICs only. |
| | SetRxEarlyThresh | (1000 1••• •••• ••••) | Set the value of the RxEarlyThresh register. |
| | SetRxFilter | (1000 0000 000• ••••) | Set the value of the RxFilter register. |
| | UpStall * | (0011 0XXX XXX0 0000)* | Stall the upload engine. |
| | UpUnStall | (0011 0XXX XXX0 0001) | Unstall the upload engine. |
| Interrupt | AcknowledgeInterrupt | (0110 1••• X••X •XX•) | Acknowledge active interrupts. |
| | RequestInterrupt | (0110 0XXX XXXX XXXX) | Cause the NIC to generate an interrupt. |
| | SetIndicationEnable | (0111 1••• •••• X••X) | Set the value of the IndicationEnable register. |
| | SetInterruptEnable | (0111 0••• •••• X••X) | Set the value of the InterruptEnable register. |
| Other | DisableDcConverter | (1011 1XXX XXXX XXXX) | Disable the 10BASE2 DC-DC converter. This command applies to 3C90xB NICs only. |
| | EnableDcConverter | (0001 0XXX XXXX XXXX) | Enable the 10BASE2 DC-DC converter. This command applies to 3C90xB NICs only. |
| | SelectRegisterWindow | (0000 1000 0000 0•••) | Change the visible window. |
| | StatisticsDisable | (1011 0XXX XXXX XXXX) | Disable collection of statistics. |
| | StatisticsEnable | (1010 1XXX XXXX XXXX) | Enable collection of statistics. |

## Reset Commands

### GlobalReset

*The GlobalReset command is not always completely executed before the next command can be issued to the NIC. The driver must ensure that the cmdInProgress bit in the IntStatus register is a zero before taking any further action with the NIC.*

| **Bit Value** | | **(0000 000• •0•• ••••)** |
|---|---|---|
| **Bit** | **Name** | **Description** |
| [0] | tpAuiReset | This bit, when set, masks reset to the 10BASE-T and AUI transceiver.<br>This bit is not supported in 3C90xB NICs with the 40-0476-001 or 04-0483-00x ASIC. |
| [1] | endecReset | This bit, when set, masks reset to the internal 10 Mbps encoder/decoder.<br>This bit is not supported in 3C90xB NICs with the 40-0476-001 or 04-0483-00x ASIC. |
| [2] | networkReset | This bit, when set, masks reset to the network interface logic, including the CSMA/CD core, and the statistics registers.<br>In 3C90xB NICs with the 40-0476-001 or 04-0483-00x ASIC, this bit masks resets to the PHY. |
| [3] | fifoReset | This bit, when set, masks reset to the FIFO control logic. |
| [4] | aismReset | This bit, when set, masks reset to the auto-initialize state machine logic.<br>If this bit is not set, the EEPROM data is reloaded. |
| [5] | hostReset | This bit, when set, masks reset to the bus interface logic.<br>If this bit is not set, the following registers are cleared: IntStatus, InterruptEnable, IndicationEnable, and Countdown. |
| [7] | vcoReset | This bit, when set, masks reset to the on-board 10 Mbps VCO.<br>This bit is not supported in 3C90xB NICs with the 40-0476-001 or 04-0483-00x ASIC. |
| [8] | upDownReset | This bit, when set, masks reset to the upload/download logic.<br>If this bit is not set, the upload and download engines are reset, including DnListPtr, UpListPtr, DmaCtrl, and UpPktStatus. |

Except for the NIC configuration aspects that are handled by the power-on self-test (POST) routines executed by the host, the NIC must be reinitialized after a GlobalReset unless the aismReset bit is set.

The registers in the PCI configuration space are not reset by the GlobalReset command, except those registers that are aliased from registers in the I/O space—InternalConfig, ResetOptions, and EepromData.

Because the NIC's serial EEPROM may need to be read as part of the reset process, this operation can take as long as 1 ms to finish. The cmdInProgress bit in IntStatus must be polled to ensure that the command has finished.

**RxReset**

*The RxReset command is not always completely executed before the next command can be issued to the NIC. The driver must ensure that the cmdInProgress bit in the IntStatus register is a zero before taking any further action with the NIC.*

| Bit Value | | (0010 100• 0000 ••••) |
|---|---|---|
| **Bit** | **Name** | **Description** |
| [0] | tpAuiRxReset | This bit, when set, masks reset to the 10BASE-T and AUI transceiver receive logic.<br>This bit is not supported in 3C90xB NICs with the 40-0476-001 or 04-0483-00x ASIC. |
| [1] | endecRxReset | This bit, when set, masks reset to the internal Ethernet encoder/decoder receive logic.<br>This bit is not supported in 3C90xB NICs with the 40-0476-001 or 04-0483-00x ASIC. |
| [2] | networkRxReset | This bit, when set, masks reset to the network interface receive logic, including the CSMA/CD core.<br>If this bit is not set, the receiver is disabled, and RxFilter is cleared.<br>In 3C90xB NICs with the 40-0476-001 or 04-0483-00x ASIC, this bit masks reset to the PHY. |
| [3] | fifoRxReset | This bit, when set, masks reset to the receive FIFO control logic.<br>If this bit is not set, the receive FIFO contents are flushed and RxEarlyThresh is set to its default (disabled) state. |
| [8] | upRxReset | This bit, when set, masks reset to the upload logic.<br>When this bit is not set, the upload logic is reset, including the UpListPtr and UpPktStatus registers, and the upComplete and upRxEarlyEnable bits in the DmaCtrl register. |

The RxReset command resets the receive logic throughout the NIC.

The 5-bit parameter acts as a bit mask, masking the reset to various portions of the receive logic.

This command should not be used after initialization except to recover from receive errors such as a receive FIFO underrun.

**TxReset**

*The TxReset command is not always completely executed before the next command can be issued to the NIC. The driver must ensure that the cmdInProgress bit in the IntStatus register is a zero before taking any further action with the NIC.*

| Bit Value | | (0101 100• 0000 ••••) |
|---|---|---|
| **Bit** | **Name** | **Description** |
| [0] | tpAuiTxReset | This bit, when set, masks reset to the 10BASE-T and AUI transceiver transmit logic.<br>This bit is not supported in 3C90xB NICs with the 40-0476-001 or 04-0483-00x ASIC. |

| Bit Value | | (0101 100• 0000 ••••) |
|---|---|---|
| **Bit** | **Name** | **Description** |
| [1] | endecTxReset | This bit, when set, masks reset to the internal Ethernet encoder/decoder transmit logic.<br>This bit is not supported in 3C90xB NICs with the 40-0476-001 or 04-0483-00x ASIC. |
| [2] | networkTxReset | This bit, when set, masks reset to the network interface transmit logic, including the CSMA/CD core.<br>If this bit is not set, the transmitter is disabled, and the TxStatus queue is cleared.<br>In 3C90xB NICs with the 40-0476-001 or 04-0483-00x ASIC, this bit masks reset to the PHY. |
| [3] | fifoTxReset | This bit, when set, masks reset to the transmit FIFO control logic.<br>If this bit is not set, the transmit FIFO is flushed, and TxStartThresh and TxReclaimThresh are forced to their default (disabled) state. |
| [8] | dnTxReset | This bit, when set, masks reset to the download logic.<br>If this bit is not set, the download logic is reset, including the DnListPtr register and the dnComplete and dnInProg bits in the DmaCtrl register. |

(2 of 2)

The TxReset command resets the transmitter logic throughout the NIC. TxReset is required after a transmit underrun or jabber error. The low-order bits mask the reset to various portions of the transmit logic.

## Transmit Commands

### DnStall

*The DnStall command is not always completely executed before the next command can be issued to the NIC. The driver must ensure that the cmdInProgress bit in the IntStatus register is a zero before taking any further action with the NIC.*

| Bit Value | (0011 0XXX XXX0 0010) |
|---|---|

The DnStall command stops the NIC from fetching the DnNextPtr DPD entry and loading it into the DnListPtr register.

If DnListPtr is nonzero, the driver must issue a DnStall command before modifying the downlist to avoid conflicts with the DnListPtr updates. The host must wait for the cmdInProgress bit to be deasserted before continuing.

3C90xB NICs do not need this command if polling is enabled.

### DnUnstall

| Bit Value | (0011 0XXX XXX0 0011) |
|---|---|

The opposite of DnStall, The DnUnstall command releases the NIC to fetch the DnNextPtr DPD entry and update the DnListPtr register. The host should issue this command as soon as possible after the DnStall command, once it has finished modifying the downlist.

### SetTxReclaimThresh

The SetTxReclaimThresh command applies to 3C90xB NICs only.

| Bit Value | (1100 0000 •••• ••••) |
|---|---|

The SetTxReclaimThresh command sets the TxReclaimThresh register to the desired value. The NIC multiplies the value by 16 before comparing it to the number of bytes transmitted.

### SetTxStartThresh

| Bit Value | (1001 1••• •••• ••••) |
|---|---|

The SetTxStartThresh command establishes the value of the TxStartThresh register. The parameter is written into bits [12:2] of TxStartThresh, and bits [1:0] are cleared.

The NIC begins transmission attempts for a packet as soon as the number of bytes downloaded to the transmit FIFO is greater than the value in TxStartThresh. If the packet being transmitted is shorter than TxStartThresh, then transmit attempts begin as soon as the entire packet has been downloaded.

### TxDisable

| Bit Value | (0101 0XXX XXXX XXXX) |
|---|---|

The TxDisable command disables the NIC's transmitter after the completion of the transmission attempt of any packet currently being transmitted.

If additional packets are queued up in the transmit FIFO, they are not transmitted. Nor are those packets discarded. If the transmitter is again enabled, packets in the transmit FIFO are transmitted.

### TxEnable

| Bit Value | (0100 1XXX XXXX XXXX) |
|---|---|

The TxEnable command enables the NIC to transmit packets.

The NIC comes out of reset with the transmitter disabled. This command must be issued before any attempt to transmit packets. The transmitter can be disabled through the use of the TxDisable or TxReset commands or by a transmitter error such as a transmit FIFO overrun.

## Receive Commands

### RxDisable

| Bit Value | (0001 1XXX XXXX XXXX) |
|---|---|

The RxDisable command prevents the NIC from receiving any further packets.

Any packet that is in the process of being received when this command is issued is not affected. This command has no effect on the contents of the receive FIFO or on any receive status or statistics.

### RxEnable

| Bit Value | (0010 0XXX XXXX XXXX) |
|---|---|

The RxEnable command enables the NIC to receive packets that meet the address-filtering requirements currently in use. If this command is issued while a packet is currently active on the network, the NIC begins reception at the beginning of the next packet.

The NIC comes out of reset with the receiver disabled. An RxEnable command must be issued to allow the NIC to receive packets. Either an RxDisable or an RxReset command can be used to disable receive operations.

### SetHashFilterBit

The SetHashFilterBit command applies to 3C90xB NICs only.

| Bit Value | (1100 1•XX XX•• ••••) |
|---|---|

This command is used to program the value of a particular bit in the hash filter for multicast packet reception. Each bit in the hash filter corresponds to a set of multicast addresses that may be received. The low-order six bits select the bit. Bit 10 is the value to be programmed into that bit.

The hash filter acts as an array of 64 enable bits (256 on 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC). Incoming packets that have the group bit set have a cyclic redundancy check (CRC) applied to their destination address.

The low-order six (or eight) bits of the CRC are used as an index into the hash filter. If the hash filter bit addressed by the index is set, the packet is accepted by the NIC and passed up to higher layers. If the hash filter bit is cleared, the packet is discarded.

*3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC and later NICs implement a 256-bit HashFilter array, requiring an 8-bit select parameter in SetHashFilterBit. It is recommended that all software compute an 8-bit CRC and use this parameter in SetHashFilterBit. An example of code that performs this computation appears below.*

```
/*
   This function takes a 6-byte Ethernet address value and
   returns the bit position in the 3C905B multicast hash filter
   that corresponds to that address. It runs the AUTODIN II CRC
   algorithm on the address value, and then returns the lower
   eight bits of the CRC.
```

```
Arguments:
    address  -  Ethernet address

Return Value:
    filter bit position
*/

unsigned short HashAddress (unsigned char Address [6])
{
    unsigned long Crc, Carry;
    unsigned int i, j;
    unsigned char ThisByte;

    /* Compute CRC for the address value. */
    Crc = 0xffffffff;  /* initial value */

    /* For each byte of the address. */
    for (i = 0; i < 6; i++)
    {
        ThisByte = Address[i];

        /* For each bit in the byte. */
        for (j = 0; j < 8; j++)
        {
           Carry = ((Crc & 0x80000000) ? 1 : 0) ^ (ThisByte & 0x01);
           Crc <<= 1;
           ThisByte >>= 1;
           if (Carry)
               Crc = (Crc ^ 0x04c11db6) | Carry;
        }
    }

    /* Return the filter bit position. */
    return Crc & 0x000000FF;
}
```

### SetRxEarlyThresh

| Bit Value | (1000 1••• •••• ••••) |
|---|---|

The SetRxEarlyThresh command sets the RxEarlyThresh register to the desired value. The parameter is written into bits [12:2] of RxEarlyThresh, and bits [1:0] are cleared.

The value in RxEarlyThresh serves two functions:

- Early receive interrupt—When the number of bytes received for a packet is greater than the value stored in RxEarlyThresh, the rxEarly indication is set.
- Upload eligibility—When the upRxEarlyEnable bit in DmaCtrl is set, RxEarlyThresh then determines the number of bytes that must be received before the packet can begin uploading.

For more information on the operation of rxEarly interrupts, see "RxEarlyThresh" in Chapter 7.

### SetRxFilter

| Bit Value | (1000 0000 000• ••••) |
|---|---|

The SetRxFilter command defines the value of the RxFilter register.

The five active parameter bits in this command may be used in any combination and are defined as follows:

| RxFilter Parameter | Addresses Enabled |
|---|---|
| XXXX1 | Individual (must match station address) |
| XXX1X | All multicast (including broadcast) |
| XX1XX | Broadcast |
| X1XXX | All (promiscuous) |
| 1XXXX (3C90xB NICs) | Multicast hash filter |

The effect of each bit is additive. That is, a 00011b pattern enables individually addressed packets that match the NIC's StationAddress as well as all multicast packets. Setting bit 3 (promiscuous mode) overrides bits [2:0].

### UpStall

*The UpStall command is not always completely executed before the next command can be issued to the NIC. The driver must ensure that the cmdInProgress bit in the IntStatus register is a zero before taking any further action with the NIC.*

| Bit Value | (0011 0XXX XXX0 0000) |
|---|---|

The UpStall command stops the NIC from fetching the UpNextPtr UPD entry and loading it into the UpListPtr register.

Whenever the host wishes to modify the uplist, and UpListPtr is nonzero, the host must issue an UpStall command to avoid conflicts with the NIC's UpListPtr updates. Note that this command requires the host to wait for the cmdInProgress bit to be deasserted before continuing.

### UpUnStall

| Bit Value | (0011 0XXX XXX0 0001) |
|---|---|

The opposite of UpStall, the UpUnStall command releases the NIC to fetch the UpNextPtr UPD entry and load the UpListPtr register. The host should issue this command as soon as possible after UpStall, once it has finished modifying the uplist.

When the upload engine stalls because it is reading a UPD that is in use (meaning the upComplete bit is set in the UpPktStatus entry in the UPD), the NIC can automatically execute an UpUnStall command by polling on the upComplete bit and waiting for the software to clear the bit. This function is enabled when the UpPoll register contains a nonzero value.

## Interrupt Commands

### AcknowledgeInterrupt

| Bit Value | (0110 1••• X••X •XX•) |
|---|---|
| **Bit** | **Name** |
| [0] | interruptLatchAck |
| [5] | rxEarlyAck |
| [6] | intRequestedAck |
| [9] | dnCompleteAck |
| [10] | upCompleteAck |

The AcknowledgeInterrupt command resets selected interrupt indications in the IntStatus register. When it is issued, the indications that correspond to bits set to one in the parameter field are cleared.

Several of the interrupt types must be acknowledged by means that are unique to the interrupt type. These means are defined in the IntStatus register definition.

Attempting to acknowledge an indication that is not active has no effect.

### RequestInterrupt

| Bit Value | (0110 0XXX XXXX XXXX) |
|---|---|

The RequestInterrupt command sets the intRequested bit in the IntStatus register (if so enabled) and causes an interrupt to the host (if so enabled).

The NIC can generate an automatic intRequested interrupt when the Countdown register count reaches zero. The driver must maintain internal state to determine what to do when an intRequested interrupt occurs.

### SetIndicationEnable

| Bit Value | (0111 1••• •••• X••X) |
|---|---|

The parameter of the SetIndicationEnable command becomes the value held in the IndicationEnable register.

Each bit corresponds to an individual indication source. See "IntStatus" in Chapter 8 for a description of the indication bits.

Indications disabled with this command do not cause an interrupt to the host, nor are they visible in the IntStatus register. The IndicationEnable register is cleared upon system reset. Bit 0 of this register is a "don't care" bit because the interruptLatch bit is always enabled.

### SetInterruptEnable

| Bit Value | (0111 0••• •••• X••X) |
|---|---|

The parameter of the SetInterruptEnable command becomes the value held in the InterruptEnable register.

Each bit corresponds to an individual interrupt source. See "IntStatus" in Chapter 8 for a description of the interrupt bits.

Interrupts disabled with this command do not cause an interrupt to the host, but they may still be set in the IntStatus register. The InterruptEnable register is cleared upon NIC reset. Bit 0 of this register is a "don't care" bit because the interruptLatch bit is always enabled.

## Other Commands

### DisableDcConverter

The DisableDcConverter command applies to the 3C905B-TX NIC only.

| Bit Value | (1011 1XXX XXXX XXXX) |
|---|---|

The DisableDcConverter command disables the DC-DC converter that drives an on-board 10BASE2 transceiver.

This command affects only 10BASE2 operation and should be used only when a NIC is so configured. The driver should wait at least 800 µs after issuing this command before attempting to use an AUI interface. The Timer register can be used to time this.

### EnableDcConverter

The EnableDcConverter command applies to the 3C905B-TX NIC with the 40-0502-00x ASIC and 3C900B NICs only.

| Bit Value | (0001 0XXX XXXX XXXX) |
|---|---|

The EnableDcConverter command enables (applies power to) the DC-DC converter that drives an on-board 10BASE2 transceiver. This command affects only 10BASE2 operation and should be used only when a NIC is so configured.

After the NIC is powered up or when it experiences a hardware reset, this command must be issued before the 10BASE2 port can be used to transmit or receive packets. The driver should wait at least 800 µs after issuing this command before attempting to transmit or receive packets. The Timer register can be used to time this.

### SelectRegisterWindow

| Bit Value | (0000 1000 0000 0•••) |
|---|---|

The SelectRegisterWindow command causes the specified register bank (windows 0 through 7) to become visible in the 16-byte register window.

The register window bank zero is the default bank upon system reset.

### StatisticsDisable

| Bit Value | (1011 0XXX XXXX XXXX) |
|---|---|

The StatisticsDisable command disables the counting of statistical events by halting the statistic counters. Disabling the counters does not alter their values.

### StatisticsEnable

| Bit Value | (1010 1XXX XXXX XXXX) |
|---|---|

The StatisticsEnable command enables the NIC's statistic counters. Upon power-up, statistics counting is disabled. This command must be issued to enable the counting of statistic events.

# 11 AUTO-NEGOTIATION AND MII REGISTERS

This chapter describes the auto-negotiation and MII registers for the 3C90xB NICs.

*The 3C90x NICs auto-negotiation function is implemented within the National Semiconductor DP83840 PHY chip. For details on that chip, refer to the DP83840 specification from National Semiconductor.*

## 3C90xB NICs Auto-Negotiation

The auto-negotiation function on the 3C90xB NICs includes several registers that allow software to read status and control various aspects of the auto-negotiation process.

These auto-negotiation registers are accessed through the MII management interface using a PHY address (PHYAD) of 11000b (shifted out left to right). Software controls the management interface directly by writing and reading bits in the NetworkDiagnostic register. For more information on programming the MII management interface, see Appendix B.

*There are three different versions of the ASIC on the 3C905B-TX NIC. These three ASICs function identically except for the PHY portion, which is proprietary to each ASIC. As a result, the MII register layouts differ on the three ASICs.*

The ASIC on the 3C905B-TX NICs can be physically identified by the following numbers:

| 3C905B-TX NIC ASIC Numbers |
|---|
| 40-0502-001<br>40-0502-002<br>40-0502-003<br>40-0502-004 |
| 40-0476-001 |
| 40-0483-001<br>40-0483-002<br>40-0483-004<br>40-0483-005 |

For more information on the ASICs, see "ASICs" in Chapter 2.

## 40-0502-00x ASIC Auto-Negotiation Registers

On 3C90xB NICs that use the 40-0502-00x ASIC, an integrated 802.3u auto-negotiation function handles auto-negotiation for 10BASE-T and 100BASE-TX media types. 100BASE-T4 is not implemented on the NICs.

The auto-negotiation function interacts with the 10 Mbps and 100BASE-X functions to negotiate a common operating mode with its link partner. If a common mode is found and a link is established, auto-negotiation directs the signal multiplexer (see Figure 23 in Chapter 4) to connect the appropriate signaling function to the MAC.

To make auto-negotiation compatible with old drivers written for external PHY devices, auto-negotiation registers are accessed through the MII management interface. However, they do not use the MII data interface. For auto-negotiation to work properly, the xcvrSelect bit in the InternalConfig register must be set to Auto-Negotiation, not to MII.

Often a physical layer device connected to the 3C90xB NIC's MII port has its own auto-negotiation function. For example, a 100BASE-T4 PHY might be able to auto-negotiate 10BASE-T and 100BASE-T4. It is important to understand that this auto-negotiation function is completely separate from the auto-negotiation function inside the 3C90xB NIC's ASIC. A driver attempting to determine the link status of a 3C90xB NIC must be prepared to check the ASIC auto-negotiation function and any auto-negotiation function contained in an MII-based device.

Table 26 summarizes the names, addresses, and functions of the 40-0502-00x ASIC auto-negotiation registers. The registers are described in alphabetical order in the section following the table.

**Table 26** Summary of 40-0502-00x ASIC Auto-Negotiation Registers

| Address | Register Name | Description |
|---|---|---|
| 00 | AutoNegControl | Contains control bits to reset, restart, and configure auto-negotiation. |
| 01 | AutoNegStatus | Contains various status and capabilities bits. |
| 02 | AutoNegPhyId1 | Contains PHY identification data. |
| 03 | AutoNegPhyId2 | Contains PHY identification data. |
| 04 | AutoNegAdvert | Controls which capabilities the NIC is allowed to advertise to the link partner. |
| 05 | AutoNegAbility | Returns the advertised abilities received from the link partner during auto-negotiation. |
| 06 | AutoNegExpansion | Contains bits pertaining to expanded auto-negotiation functions. |

### AutoNegAbility

The AutoNegAbility register contains the advertised capabilities of the link partner, as received during the auto-negotiation process.

This register resets to 0000h.

**AutoNegAbility Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 0 | 0 | 0 | | | | | | | | | | |

**AutoNegAbility Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [4:0] | protSelect | This read-only field contains the link partner's encoded protocol selector. |
| [5] | baseT | This read-only bit indicates that the link partner supports 10BASE-T. |
| [6] | baseTFullDuplex | This read-only bit indicates that the link partner supports 10BASE-T full-duplex. |
| [7] | baseTx | This read-only bit indicates that the link partner supports 100BASE-TX. |
| [8] | baseTxFullDuplex | This read-only bit indicates that the link partner supports 100BASE-TX full-duplex. |
| [9] | baseT4 | This read-only bit indicates that the link partner supports 100BASE-T4. This bit always returns zero. |
| [13] | remoteFault | This read-only bit indicates that the link partner detected a remote fault. |
| [14] | acknowledge | This read-only bit indicates that the link partner has acknowledged receipt of the 3Com ability data word. |
| [15] | nextPage | This read-only bit, when set, indicates that the link partner is next-page capable. |

## AutoNegAdvert

The AutoNegAdvert register contains the capabilities that the NIC advertises when it is auto-negotiating.

**AutoNegAdvert Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | 0 | 0 | 0 | 0 | | | | | | | | | |

**AutoNegAdvert Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0:4] | protSelect | These read/write bits indicate which protocol is supported. These bits default at reset to 1h, which indicates a CSMA/CD device. |
| [5] | baseT | This read/write bit indicates 10BASE-T support is advertised. This bit resets to one.<br>Software may write a new value to this bit and restart auto-negotiation to advertise a new set of capabilities. |
| [6] | baseTFullDuplex | This read/write bit indicates that 10BASE-T full-duplex support is advertised. This bit resets to one.<br>Software may write a new value to this bit and restart auto-negotiation to advertise a new set of capabilities.<br>Also, if the disableAdvFD bit in the ResetOptions register is set, this bit will be zero regardless of any writes to AutoNegAdvert. |

(1 of 2)

**AutoNegAdvert Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [7] | baseTx | This read/write bit indicates that 100BASE-TX support is advertised. This bit resets to one.<br>Software may write a new value to this bit and restart auto-negotiation to advertise a new set of capabilities.<br>Also, if the disableAdv100 bit in the ResetOptions register is set, this bit is zero regardless of any writes to AutoNegAdvert.<br>This bit is not implemented in 3C900B NICs. It always returns zero. |
| [8] | baseTxFullDuplex | This read/write bit indicates that 100BASE-TX full-duplex support is advertised. This bit resets to one.<br>Software may write a new value to this bit and restart auto-negotiation to advertise a new set of capabilities.<br>Also, if the disableAdvFD or disableAdv100 bit in the ResetOptions register is set, this bit is zero regardless of any writes to AutoNegAdvert.<br>This bit is not implemented in 3C900B NICs. It always returns zero. |
| [13] | remoteFault | This read/write bit determines what value is advertised in the remote fault bit position. This bit resets to zero.<br>Upon detection of a remote fault, software may set this bit, and if necessary, restart auto-negotiation. |
| [14] | acknowledge | This read-only bit is set when the NIC has received the link code word from the link partner. This bit stays set until auto-negotiation is restarted or the NIC is reset. |

(2 of 2)

## AutoNegControl

The AutoNegControl register resets to 3100h unless the disableAutoNeg bit in the ResetOptions register is set, in which case the reset value is 2100h.

**AutoNegControl Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | | | 0 | 0 | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**AutoNegControl Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [8] | fullDuplex | This bit has no function.<br>To configure the NIC for full-duplex operation, regardless of which media port is selected, set the fullDuplexEnable bit in the MacControl register. |
| [9] | restartAutoNeg | Setting this bit restarts the auto-negotiation process.<br>This bit has no meaning when the autoNegEnable bit in this register is cleared.<br>This bit is self-clearing and returns a value of one until the auto-negotiation process is initiated. |

(1 of 2)

**AutoNegControl Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [12] | autoNegEnable | The value of this read/write bit determines whether the auto-negotiation logic is enabled to perform auto-negotiation.<br><br>This bit resets to one (enabled). However, if the disableAutoNeg bit in the ResetOptions register is set or if the xcvrSelect bit in the InternalConfig register is set to any setting except Auto-Negotiation, this bit always returns zero, regardless of any writes to this bit. |
| [13] | 100Enable | Setting this read/write bit configures the auto-negotiation logic for 100 Mbps operation. Clearing it selects 10 Mbps operation.<br><br>This bit is ignored by auto-negotiation if the autoNegEnable bit in this register is set.<br><br>This bit sets the operating mode for the auto-negotiation logic when the auto-negotiation process is disabled. Additionally, this bit has no meaning when the xcvrSelect bit in the InternalConfig register is set to any setting except Auto-Negotiation.<br><br>This bit is not implemented in 3C900B NICs. It always returns zero. |
| [15] | autoNegReset | Setting this read/write bit resets the auto-negotiation logic and returns the auto-negotiation registers to their reset default values.<br><br>When this bit is set, it returns a value of one until the reset sequence is complete, at which time the bit is automatically cleared. |

(2 of 2)

## AutoNegExpansion

The AutoNegExpansion register contains some assorted bits related to auto-negotiation.

**AutoNegExpansion Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | 0 | | |

**AutoNegExpansion Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | lpAutoNegAble | This read-only bit indicates that the link partner supports auto-negotiation. |
| [1] | pageReceived | This read-only bit is set when a new link code word page has been received from the link partner. This bit is automatically cleared when the AutoNegExpansion register is read. |
| [3] | lpNextPageAble | This read-only bit, when set, indicates that the link partner supports next page. |

(1 of 2)

**AutoNegExpansion Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [4] | parallelDetectFault | When set, this read-only bit indicates that the link was left unresolved because a fault was discovered by the parallel detection logic. This bit indicates one of two conditions:<br>■ Neither the 10BASE-T link integrity function nor the 100BASE-TX link integrity function indicated a valid link.<br>■ Both the 10BASE-T link integrity function and the 100BASE-TX link integrity function indicated a valid link.<br>This bit is cleared when the AutoNegExpansion register is read. |
| [14] | linkFailInt | This read-only bit is set when the auto-negotiation logic leaves the LINK GOOD state as a result of link failure.<br>Reading AutoNegExpansion clears this bit. When this bit is set, a linkEvent interrupt occurs if the linkEvent bit in the IntStatus register is enabled. |
| [15] | linkGoodInt | This read-only bit is set when the auto-negotiation logic enters the LINK GOOD state as a result of successful auto-negotiation.<br>Reading AutoNegExpansion clears this bit. When this bit is set, a linkEvent interrupt occurs if linkEvent is enabled in the IntStatus register. |

(2 of 2)

### AutoNegPhyId1 and AutoNegPhyId2

These registers provide PHY identification values to management software. Because the auto-negotiation logic is contained within the ASIC, there is no special need to provide a PHYID; these registers return zero, as allowed in the IEEE 802.3u specification.

### AutoNegStatus

The AutoNegStatus register resets to 7809h (1809h on 3C900B NICs) unless the disableAutoNeg bit in the ResetOptions register is set, in which case the reset value is 7801h (1801h on 3C900B NICs).

**AutoNegStatus Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | 0 | 0 | 0 | 0 | | | | | | | |

**AutoNegStatus Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | extendedCapable | This read-only bit always returns one, indicating that auto-negotiation implements the MII extended register set. |
| [1] | jabberDetect | This bit always returns zero. The NIC implements jabber indication in the TxStatus register. |

(1 of 2)

**AutoNegStatus Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [2] | linkStatus | This read-only bit indicates that a valid link has been established for either 10 Mbps or 100 Mbps operation.<br><br>This bit is implemented with a latching function: any occurrence of a link failure condition causes this bit to be cleared until AutoNegStatus is read. |
| [3] | autoNegAble | This read-only bit indicates that the NIC is able to perform auto-negotiation.<br><br>This bit is the logical opposite of the disableAutoNeg bit in the ResetOptions register. |
| [4] | remoteFault | This read-only bit, when set, indicates that a remote fault condition has been detected. This bit is cleared by reset or by reading AutoNegStatus. |
| [5] | autoNegComplete | This read-only bit is set when the auto-negotiation process is complete.<br><br>If auto-negotiation is never able to establish a link, then this bit stays clear indefinitely. The driver must provide a timeout loop that determines when to give up on the auto-negotiation process. |
| [6] | prmbleSuppress | This read-only bit is permanently zero, to indicate that accesses to the auto-negotiation registers require a preamble. |
| [11] | baseTHalfDuplex | This read-only bit always returns one, indicating auto-negotiation's ability to negotiate a 10BASE-T half-duplex link. |
| [12] | baseTFullDuplex | This read-only bit always returns one, indicating auto-negotiation's ability to negotiate a 10BASE-T full-duplex link. |
| [13] | baseTxHalfDuplex | This read-only bit always returns one, indicating auto-negotiation's ability to negotiate a 100BASE-TX half-duplex link.<br><br>This bit is not implemented in 3C900B NICs. It always returns zero. |
| [14] | baseTxFullDuplex | This read-only bit always returns one, indicating auto-negotiation's ability to negotiate a 100BASE-TX full-duplex link.<br><br>This bit is not implemented in 3C900B NICs. It always returns zero. |

(2 of 2)

## 40-0476-001 ASIC Auto-Negotiation Registers

On 3C90xB NICs that use the 40-0476-001 ASIC, an integrated 802.3u auto-negotiation function handles auto-negotiation for 10BASE-T and 100BASE-TX media types. 100BASE-T4 is not implemented on the NICs.

The 40-0476-001 ASIC contains the ability to negotiate its mode of operation over the twisted-pair link using the auto-negotiation mechanism defined in the IEEE 802.3u specification.

Auto-negotiation may be enabled or disabled by hardware or software control. When the auto-negotiation function is enabled, the NIC automatically chooses its mode of operation by advertising its abilities and comparing them with those received from its link partner.

The 40-0476-001 ASIC can be configured to advertise 100BASE-TX full-duplex and half-duplex and 10BASE-T full-duplex and half-duplex. Each transceiver negotiates independently with its link partner and chooses the highest level of operation available for its own link.

Auto-negotiation is not operational during 100BASE-FX operation and does not advertise full-duplex abilities when the device is configured in repeater mode.

Table 27 summarizes the names, addresses, and functions of the 40-0476-001 ASIC auto-negotiation and MII registers. The registers are described in alphabetical order in the sections following the table.

**Table 27** Summary of 40-0476-001 ASIC Auto-Negotiation and MII Registers

| Address | Register Name | Description |
|---|---|---|
| 00H | Control | Contains control bits to reset, restart, and configure auto-negotiation. |
| 01h | Status | Contains various status and capabilities bits. |
| 02h | PHYID High | Contains PHY identification data. |
| 03h | PHYID Low | Contains PHY identification data. |
| 04h | Auto-Negotiation Advertise | Controls which capabilities the NIC is allowed to advertise to the link partner. |
| 05h | Link Partner Ability | Returns the advertised abilities received from the link partner during auto-negotiation. |
| 06h | Auto-Negotiation Expansion | Contains bits pertaining to expanded auto-negotiation functions. |
| 07h | Next Page | Not implemented; this register is not accessible. |
| 10h | 100BASE-X Auxiliary Control | See the register bit definitions for a description of this register. |
| 11h | 100BASE-X Auxiliary Status | See the register bit definitions for a description of this register. |
| 12h | 100BASE-X Receive Error Counter | Contains bits that increment each time the NIC receives a noncollision packet containing at least one receive error. |
| 13h | 100BASE-X False Carrier Counter | Contains bits that increment each time the NIC detects a false carrier on the receive input. |
| 14h | 100BASE-X Disconnect Counter | Contains bits that increment each time the carrier integrity monitor within the NIC enters the link unstable state. |
| 15h | TX Equalizer Coefficient Read/Write | See the register bit definitions for a description of this register. |

(1 of 2)

**Table 27** Summary of 40-0476-001 ASIC Auto-Negotiation and MII Registers (continued)

| Address | Register Name | Description |
|---|---|---|
| 16h | TX Equalizer Coefficient Control | See the register bit definitions for a description of this register. |
| 18h | Auxiliary Control/Status | See the register bit definitions for a description of this register. |
| 19h | Auxiliary Status Summary | Contains redundant status bits found elsewhere within the MII register space. |
| 1Ch | 10BASE-T Auxiliary Error and General Status | Returns auxiliary errors and general status implemented in 10BASE-T mode. |
| 1Dh | Auxiliary Mode | See the register bit definitions for a description of this register. |
| 1Eh | Auxiliary Multiple-PHY | See the register bit definitions for a description of this register. |

(2 of 2)

## 10BASE-T Auxiliary Error and General Status

The 10BASE-T Auxiliary Error and General Status register contains read-only bits that are latched high. When certain types of errors occur in the ASIC, one or more corresponding error bits are set to one. They remain so until the register is read, or until a chip reset occurs. All such errors necessarily result in data errors, and are indicated by a high value on the RXER output pin at the time the error occurs.

**10BASE-T Auxiliary Error and General Status Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | |

**10BASE-T Auxiliary Error and General Status Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | full-duplex indication | This read-only bit returns a one when the NIC is in full-duplex mode.<br>In all other modes, the bit returns a zero. |
| [1] | speed indication | This read-only bit indicates the current operation speed of the NIC.<br>■ 1 = 100BASE-X operation<br>■ 0 = 10BASE-T operation<br>While the auto-negotiation exchanged is performed, the NIC is always operating at 10BASE-T speed. |
| [2] | force 100/10 indication | This read-only bit returns a one when the speed is forced to 100BASE-X. It returns a zero when the speed is forced to 10BASE-T. A value of zero is returned when one of the following instances is true:<br>■ The ANEN pin is low and the F100 pin is low.<br>■ The auto-negotiation enable bit (12) and the forced speed selection bit (13) in the Control register are set to zero. |

(1 of 2)

**10BASE-T Auxiliary Error and General Status Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [3] | auto-negotiation indicator | This bit, when set to one, indicates that auto-negotiation is activated.<br>A combination of a one in the auto-negotiation enable bit (12) in the Control register and a logic one on the ANEN input pin is required to enable auto-negotiation.<br>When set to zero, this bit indicates that the speed is manually forced. |
| [4] | repeater mode indication | This bit returns a one when the NIC is in repeater mode. It returns a zero when the NIC is in DTE mode.<br>This bit returns the same value as the RPTMODE input pin. |
| [5:7] | revision | These read-only bits return the revision number of the ASIC. The current revision is labeled 000. |
| [8] | polarity error | This bit returns a one whenever the polarity of the receive channel is inverted. It returns a zero when the polarity of the receive channel is correct.<br>The NIC is capable of automatically inverting the polarity of the receive channel. No data errors are reported to indicate that the automatic polarity inversion is occurring. |
| [9] | EOF error | This bit returns a one when the end of frame (EOF) sequence is improperly received.<br>This bit is valid during 10BASE-T operation only. |
| [10] | Manchester code error | This bit returns a one when a Manchester code violation is received.<br>This bit is valid during 10BASE-T operation only. |

(2 of 2)

## 100BASE-X Auxiliary Control

**100BASE-X Auxiliary Control Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | | 0 | 0 | | | | | | | 0 | 0 | 0 | 0 | 0 |

**100BASE-X Auxiliary Control Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [5] | FEF enable | This bit, when set to one, enables the far end fault (FEF) mechanism associated with 100BASE-FX operation. When set to zero, it disables the FEF mechanism. |
| [6] | baseline wander correction disable | This bit, when set to one, disables the baseline wander correction circuit. The NIC corrects the baseline wander on the receive data signal when this bit is cleared. |
| [7] | bypass receive symbol alignment | This bit, when set to one, bypasses the receive symbol alignment.<br>When used in conjunction with the bypass 4B5B encoder/decoder in this register, unaligned 5B codes are placed directly on the RXER and RXD [3:0] pins.<br>This bit defaults to the value on the SEL5B mode pin at reset. |

(1 of 2)

**100BASE-X Auxiliary Control Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [8] | bypass MLT3 encoder/decoder | This bit, when set to one, bypasses the MLT3 encoder and decoder. NRX data is transmitted and received on the cable. When set to zero, this bit enables the MLT3 encoder. |
| [9] | bypass scrambler/descrambler | This bit, when set to one, disables the stream cipher function. When set to zero, this bit enables the stream cipher function. |
| [10] | bypass 4B5B encoder/decoder | This bit, when set to one, bypasses the 4B5B encoder and decoder.<br><br>The transmitter sends 5B codes from the TXER and TXD [3:0] pins directly to the scrambler. TXEN is ignored and frame encapsulation (insertion of J/K and T/R codes) is not performed.<br><br>The receiver places descrambled and aligned 5B codes onto the RXER and RXD [3:0] pins. CRS is still asserted when a valid frame is received.<br><br>This bit defaults to the value on the SEL5B pin at reset. |
| [13] | transmit disable | This bit, when set to one, disables the transmitter. The transmitter output (TD±) is forced into a high impedance state. |

(2 of 2)

## 100BASE-X Auxiliary Status

**100BASE-X Auxiliary Status Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | | | | | 0 | | | | | | |

**100BASE-X Auxiliary Status Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | MLT3 code error detected | This bit returns a one if an MLT3 coding error is detected in the receive data stream since the last time this register was read; otherwise, it returns a zero. |
| [1] | lock error detected | This bit returns a one if the descrambler has lost lock since the last time this register was read; otherwise, it returns a zero. |
| [2] | transmit error detected | This bit returns a one if a packet is received with a transmit error code since the last time this register was read; otherwise, it returns a zero. |
| [3] | receive error detected | This bit returns a one if a packet was received with an invalid code since the last time this register was read; otherwise, it returns a zero. |
| [4] | bad ESD detected | This bit returns a one if a bad end of stream error has been detected since the last time the register was read; otherwise, it returns a zero. |
| [5] | false carrier detected | This bit returns a one if a false carrier is detected since the last time the register was read; otherwise, it returns a zero. |
| [7] | remote fault | This bit returns a one while its link partner is signaling a far-end fault condition; otherwise, it returns a zero. |

(1 of 2)

**100BASE-X Auxiliary Status Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [8] | link status | This bit returns a one when the link status is good; otherwise, it returns a zero. |
| [9] | locked | This bit returns a one when the descrambler is locked to the incoming data stream; otherwise, it returns a zero. |
| [10] | FX mode | This bit returns a one when SD± input pins are driven with a valid differential signal level. It returns a zero when both SD± and SD– are simultaneously driven low. |

(2 of 2)

### 100BASE-X Disconnect Counter

The 100BASE-X Disconnect Counter register increments each time the carrier integrity monitor within the NIC enters the link unstable state. This register automatically clears itself when read.

**100BASE-X Disconnect Counter Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | |

**100BASE-X Disconnect Counter Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [7:0] | disconnect counter | These bits indicate the number of disconnects since the last read. |

### 100BASE-X False Carrier Sense Counter

The 100BASE-X False Sense Counter register increments each time the NIC detects a false carrier on the receive input. This register automatically clears itself when read.

**100BASE-X False Carrier Sense Counter Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | |

**100BASE-X False Carrier Sense Counter Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [7:0] | false carrier sense counter | These bits indicate the number of false carrier sense events since the last read. |

### 100BASE-X Receive Error Counter

The 100BASE-X Receive Error Counter register increments each time the NIC receives a noncollision packet containing at least one receive error. This counter automatically clears itself when read.

**100BASE-X Receive Error Counter Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | |

**100BASE-X Receive Error Counter Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [7:0] | receive error counter | These bits indicate the number of noncollision packets with receive errors since the last read. |

## Auto-Negotiation Advertise

The Auto-Negotiation Advertise register controls which capabilities the NIC is allowed to advertise to the link partner.

**Auto-Negotiation Advertise Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | | 0 | 0 | | | | | | | | | | | |

**Auto-Negotiation Advertise Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [4:0] | advertise selector field | These bits contain the fixed value 00001, indicating that the chip belongs to the IEEE 802.3 class of PHY transceivers. |
| [5] | advertise 10BASE-T | This bit, when set to one, transmits 10BASE-T functionality to the link partner. When set to zero, 10BASE-T functionality is not transmitted.<br>The default value reflects the abilities of the NIC. Resetting the NIC restores the default value.<br>Reading the register returns either the value that was last written to the bit or the default value (if no write has been completed since the last reset). |
| [6] | advertise 10BASE-T FDX | This bit, when set to one, transmits 10BASE-T full-duplex functionality to the link partner. When set to zero, 10BASE-T full-duplex functionality is not transmitted.<br>The default value reflects the abilities of the NIC. Resetting the NIC restores the default value.<br>Reading the register returns either the value that was last written to the bit or the default value (if no write has been completed since the last reset). |
| [7] | advertise 100BASE-X | This bit, when set to one, transmits 100BASE-X functionality to the link partner. When set to zero, 100BASE-X functionality is not transmitted.<br>The default value reflects the abilities of the NIC. Resetting the NIC restores the default value.<br>Reading the register returns either the value that was last written to the bit or the default value (if no write has been completed since the last reset). |
| [8] | advertise 100BASE-X FDX | This bit, when set to one, transmits 100BASE-X full-duplex functionality to the link partner. When set to zero, 100BASE-X full-duplex functionality is not transmitted.<br>The default value reflects the abilities of the NIC. Resetting the NIC restores the default value.<br>Reading the register returns either the value that was last written to the bit or the default value (if no write has been completed since the last reset). |

**Auto-Negotiation Advertise Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [9] | advertise 100BASE-T4 | This bit, when set to one, transmits 100BASE-T4 functionality to the link partner. When set to zero, 100BASE-T4 functionality is not transmitted.<br>The default value reflects the abilities of the NIC. Resetting the NIC restores the default value.<br>Reading the register returns either the value that was last written to the bit or the default value (if no write has been completed since the last reset). |
| [10] | pause operation | This bit, when set to one, indicates the availability of additional DTE capability when full-duplex operation is in use.<br>The use of this bit is orthogonal to the negotiated data rate, medium, or link technology. |
| [13] | remote fault | This bit, when set to one, causes a remote fault indicator to be sent to the link partner during auto-negotiation. When set to zero, this bit clears the remote fault transmission. Resetting the NIC also clears the remote fault transmission.<br>This bit returns the value that was last written to it. If no write has been complete since the last chip reset, a zero is returned. |

(2 of 2)

## Auto-Negotiation Expansion

The Auto-Negotiation Expansion register contains bits pertaining to expanded auto-negotiation functions.

**Auto-Negotiation Expansion Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | 0 | | |

**Auto-Negotiation Expansion Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | link partner auto-negotiation able | This bit returns a one when the link partner is known to have auto-negotiation capability.<br>The bit returns a zero before any auto-negotiation information is exchanged or if the link partner does not comply with the IEEE auto-negotiation specification. |
| [1] | page received | This bit is latched high when a new link code word is received from the link partner, checked, and acknowledged. It remains high until the register is read or until the chip is reset. |
| [3] | link partner next page able | This bit returns a one when the link partner has next page capabilities. |
| [4] | parallel detection fault | This read-only bit is latched high when a parallel detection fault occurs in the auto-negotiation state machine. For more details, see the IEEE 802.3 specification.<br>This bit is reset to zero after the register is read, or when the NIC is reset. |

## Auxiliary Control/Status

**Auxiliary Control/Status Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 0 | 0 | 0 | 0 | | | | | | | | | |

**Auxiliary Control/Status Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | full-duplex indication | This bit, when set to one, indicates that full-duplex operation is active. When set to zero, full-duplex operation is not active. |
| [1] | speed indication | This read-only bit returns one when the NIC is in 100BASE-X operation. It returns zero when it is in 10BASE-T operation.<br><br>When auto-negotiation exchange is performed, the NIC is always operating at 10BASE-T speed. |
| [2] | force 100/10 indication | This read-only bit returns a one when the speed is forced to 100BASE-X operation. It returns a zero when one of the following instances is true:<br><br>■ The ANEN pin is low and the F100 pin is low.<br><br>■ The auto-negotiation enable bit (12) of the Control register is set to zero.<br><br>In all other instances, either the speed is not forced (auto-negotiation is enabled) or the speed is forced to 100BASE-X. |
| [3] | auto-negotiation indicator | This read-only bit indicates whether auto-negotiation has been enabled or disabled on the NIC.<br><br>A combination of a one in the auto-negotiation enable bit (12) in the Control register and a logic one on the ANEN input pin is required to enable auto-negotiation. |
| [4:5] | edge rate | These control bits are used to program the transmit DAC output edge rate in 100BASE-TX mode.<br><br>These bits are logically ANDed with the ER[1:0] input pins to produce the internal edge-rate controls (Edge_Rate_1 AND ER1, Edge_Rate_0 AND ER0).<br><br>■ 00 = 1 ns<br><br>■ 01 = 2 ns<br><br>■ 10 = 3 ns<br><br>■ 11 = 4 ns |
| [6:7] | HSQ:LSQ | These bits extend or decrease the squelch levels for detection of incoming data packets.<br><br>The default squelch levels implemented are those defined in the IEEE standard. The high- and low-squelch levels are useful for situations where the IEEE-prescribed levels are inadequate. The squelch levels are used by the CRS/LNK block to filter out noise and recognize only valid packet preambles and link integrity pulses.<br><br>Extending the squelch levels allows the NIC to operate properly over longer cable lengths. Decreasing the squelch levels may be useful in situations where a high level of noise is present on the cables. Reading these two bits returns the value of the squelch levels. |

(1 of 2)

**Auxiliary Control/Status Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [8] | test mode | This bit, when set to one, enables the active-high test mode. When set to zero, it enables normal operation. |
| [14] | link disable | This bit, when set to one, disables the link integrity state machines and places the NIC into forced link pass status. When set to zero, this bit restores the link integrity functions.<br>Resetting the chip also restores the link integrity functions.<br>Reading this bit returns the value of the link integrity disable status. |
| [15] | jabber disable | This bit, when set to one, disables the jabber detect function, as defined in the IEEE standard.<br>This function shuts off the transmitter when a transmission request has exceeded a maximum time limit.<br>When this bit is set to zero, or when the chip is reset, normal operation is enabled.<br>Reading this bit returns the value of the jabber detect disable status. This bit is implemented in 10BASE-T mode only. |

(2 of 2)

## Auxiliary Mode

**Auxiliary Mode Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | 0 | | |

**Auxiliary Mode Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | RXER code mode | This bit, when set to one, enables the RXER code mode.<br>In RXER code mode, when a receive data error occurs (indicated by pins RXDV=1 and RXER=1), the RXC [3:0] bus will contain a nonzero, four-bit encoded value indicating the type of error.<br>When set to zero, or when the chip is reset, normal operation is enabled.<br>This mode does not disrupt normal communication with the MAC layer and can safely be used at all times. The RXER code mode is not available in 10BASE-T serial mode. |
| [1] | 10BT serial mode | This bit, when set to one, enables the 10BASE-T serial mode. When set to zero, it disables the 10BASE-T serial mode.<br>In serial mode, data packets traverse to the MAC layer across TXD[0] and RXD[0] at a rate of 10 MHz. This bit provides the only method to place the NIC into 10BASE-T serial mode.<br>When set to zero, or when the chip is reset, MII-compliant parallel 10BASE-T operation is enabled. Serial operation is not available in 100BASE-X mode. |
| [3] | link LED disable | This bit, when set to one, disables the LINK LED output pin. When set to zero, the LINK LED output pin is enabled. |

(1 of 2)

**Auxiliary Mode Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [4] | activity LED disable | This bit, when set to one, disables the XMTLED# and RCVLED# output pins. When set to zero, the XMTLED# and RCVLED# output pins are enabled. |
| [12] | segmentation enable | This bit, when set to one, enables segmentation for this MII interface. When set to zero, the PHY connects to its default MII interface, as specified by the default settings of the segmentation control bit. |
| [13] | PHY enable | This bit, when set to one, connects the PHY to the MII interface specified in the segmentation control bit. When set to zero, the PHY is disconnected from the selected MII interface.<br>This bit is applicable only when the segmentation enable bit is active. |
| [14:15] | segmentation control | These bits select the MII interface to which the PHY is connected.<br>■ 00 = PHY connected to MII I/F 1<br>■ 01 = PHY connected to MII I/F 2<br>■ 10 = PHY connected to MII I/F 3<br>■ 11 = PHY connected to MII I/F 4 |

(2 of 2)

## Auxiliary Multiple PHY

**Auxiliary Multiple PHY Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 0 | 0 | | | | | | | 0 | 0 | 0 |

**Auxiliary Multiple PHY Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [3] | super isolate | This bit, when set to one, places the NIC in super isolate mode. When set to zero, it enables normal operation.<br>The super isolate mode is similar to the isolate mode. All MII inputs are ignored and all MII outputs are tri-stated. Additionally, all link pulses are suppressed and not transmitted. This allows the NIC to coexist with another PHY on the same NIC, with only one PHY being activated at any time. |
| [4] | ability detect | This read-only bit returns a one when the auto-negotiation state machine is in the ability detect state. This bit returns a zero when the auto-negotiation state machine is not in the ability detect state.<br>The ability detect state is entered after the auto-negotiation process begins. This state is exited after the first FLP burst or link pulses are detected from the link partner. |
| [5] | acknowledge detected | This read-only bit is latched high when the arbitrator state machine exits the acknowledge detected state. It remains high until the auto-negotiation process is restarted or the NIC is reset. |

(1 of 2)

**Auxiliary Multiple PHY Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [6] | acknowledge complete | This read-only bit returns a one after the acknowledgment exchange portion of the auto-negotiation process is complete and the arbitrator state machine has exited the acknowledge complete state.<br><br>It remains this value until the auto-negotiation process is restarted, a link fault occurs, auto-negotiation is disabled, or the NIC is reset. |
| [7] | auto-negotiation complete | This read-only bit returns a one after the auto-negotiation process is complete. It remains this value until the auto-negotiation process is restarted.<br><br>This bit returns a zero if the auto-negotiation process is still in progress or if auto-negotiation is disabled. |
| [8] | restart auto-negotiation | This self-clearing bit, when set to one, restarts auto-negotiation, regardless of the current status of the state machine.<br><br>Because this bit is self-clearing after only a few cycles, it always returns a zero when read. The operation of this bit is identical to the restart auto-negotiation bit (9) in the Control register.<br><br>Auto-negotiation must be enabled for this bit to work. |
| [11] | HCD_10BASE-T | This read-only bit reports that the highest common denominator (HCD) result of the auto-negotiation process is 10BASE-T. |
| [12] | HCD_10BASE-T_FDX | This read-only bit reports that the highest common denominator (HCD) result of the auto-negotiation process is 10BASE-T full-duplex. |
| [13] | HCD_TX | This read-only bit reports that the highest common denominator (HCD) result of the auto-negotiation process is 100BASE-TX. |
| [14] | HCD_T4 | This read-only bit reports that the highest common denominator (HCD) result of the auto-negotiation process is 100BASE-T4. |
| 15 | HCD_TX_FDX | This read-only bit reports that the highest common denominator (HCD) result of the auto-negotiation process is 100BASE-TX full-duplex. |

(2 of 2)

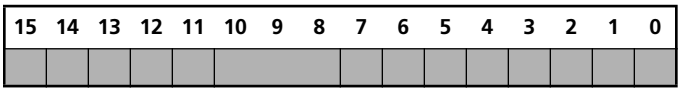
Immediately after the first link pass state (after each reset or restart auto-negotiation) only one of bits [11:15] of this register returns a one. The link pass state is identified by a one on the acknowledge complete bit or auto-negotiation complete bit in this register.

Bits [11:15] of this register are reset to zero each time auto-negotiation is restarted or when the NIC is reset. For their intended application, these bits uniquely identify the HCD only after the first link pass from reset to restart of auto-negotiation. On later link fault and subsequent auto-negotiations, if the ability of the link partner is different, then more than one of the above bits is active.

## Auxiliary Status Summary

The Auxiliary Status Summary register contains redundant status bits found elsewhere within the MII register space.

**Auxiliary Status Summary Register Format**




**Auxiliary Status Summary Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | jabber detect | This bit returns a one if a jabber condition has been detected. After the bit is read once, or if the NIC is reset, it returns a zero.<br><br>This bit is implemented in 10BASE-T operation only. |
| [1] | auto-negotiation enabled | Auto-negotiation can be disabled by one of two methods: hardware or software control.<br><br>If the ANEN input pin is driven to a logic zero, auto-negotiation is disabled by hardware control. If the auto-negotiation enabled bit is written with a value of zero, auto-negotiation is disabled by software control.<br><br>When auto-negotiation is disabled by software control, writing a one to this bit or resetting the NIC enables auto-negotiation.<br><br>Writing to this bit has no effect when auto-negotiation is disabled by hardware control. When read, this bit returns the value most recently written to it, or a one if it has not been written to since the last NIC reset. |
| [2] | link status | This bit returns a one when the link state machine is in link pass state, indicating that a valid link has been established. If a link has not been established, this bit returns a zero.<br><br>When a link failure occurs after the link pass state has been entered, the link status bit latches at zero and remains so until the bit is read. After the bit is read, it becomes a one when the link pass state is entered again. |
| [3] | speed indication | This read-only bit, when set to one, indicates 100BASE-X operation. When set to zero, this bit indicates 10BASE-T operation.<br><br>When auto-negotiation exchange is performed, the NIC is always operating at 10BASE-T speed. |
| [4] | link partner auto-negotiation able | This bit returns a one when the link partner is known to have auto-negotiation capability. The bit returns a zero before any auto-negotiation information is exchanged or if the link partner does not comply with the IEEE auto-negotiation specification. |
| [5] | link partner page received | This bit returns a one when a new page has been received. |
| [6] | link partner remote fault | This bit returns a one while its link partner is signaling a far-end fault condition; otherwise, it returns a zero. |
| [7] | auto-negotiation parallel detection fault | This bit returns a one when an auto-negotiation parallel detection fault is detected; otherwise, it returns a zero. |

(1 of 2)

**Auxiliary Status Summary Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [8:10] | auto-negotiation HCD | This bit indicates the highest common denominator (HCD) discovered with auto-negotiation.<br>■ 000 = No highest common denominator<br>■ 001 = 10BASE-T<br>■ 010 = 10BASE-T full-duplex<br>■ 011 = 100BASE-TX<br>■ 100 = 100BASE-T4<br>■ 101 = 100BASE-TX full-duplex<br>■ 11x = Undefined |
| [11] | auto-negotiation pause | This bit, when set to one, indicates the availability of additional DTE capability when full-duplex operation is in use. The use of this bit is orthogonal to the negotiated data rate, medium, or link technology. |
| [12] | auto-negotiation ability detect | This bit, when set to one, indicates auto-negotiation for link partner ability. |
| [13] | auto-negotiation acknowledge detected | This bit, when set to one, indicates that an auto-negotiation acknowledge state is detected. |
| [14] | auto-negotiation complete acknowledge | This bit, when set to one, indicates a completed acknowledge state. |
| [15] | auto-negotiation complete | This bit returns a one if the auto-negotiation process is complete and the contents of the Auto-Negotiation Advertise, Link Partner Ability, and Auto-Negotiation Expansion registers are valid. The bit returns a zero if the auto-negotiation process is not complete. |

(2 of 2)

## Control

The Control register contains control bits to reset, restart, and configure auto-negotiation. The reset value is 3000h.

**Control Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Control Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [7] | collision test enable | This bit, when set to one, enables the collision test mode. When set to zero, it disables the collision test mode.<br>Resetting the NIC also disables the collision test mode. This bit should only be set while the NIC is in loopback test mode. The COL pin may be tested by activating the Collision Test mode. While the NIC is in this mode, asserting TXEN causes the COL output to go high within 512 bit times. Deasserting TXEN causes the COL output to go low within 4 bit times. |

(1 of 3)

**Control Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [8] | duplex mode | When this bit is set to one and the auto-negotiation enable bit (12) in this register is set to zero, the NIC is forced to full-duplex mode. When set to zero or when the NIC is reset, the NIC is forced to half-duplex mode.<br><br>The default setting is half-duplex mode. |
| [9] | restart auto-negotiation | This bit, when set to one, restarts the auto-negotiation process, regardless of the current status of the auto-negotiation state machine. When set to zero, this bit has no effect.<br><br>For this bit to have an effect, auto-negotiation must be enabled. Because this bit is self-clearing after only a few cycles, it always returns a zero when read. The operation of this bit is identical to that of bit 8 in the Auxiliary Multiple PHY register. |
| [10] | isolate | This bit, when set to one, isolates the NIC from its Media Independent Interface (MII). All MII outputs are tri-stated and are ignored.<br><br>When set to zero, this bit clears the isolate mode because the MII management interface is still active. The isolate mode can also be cleared by resetting the NIC.<br><br>When this bit is read and the block is in isolate mode, it returns a one. When this bit is read and the block is not in isolate mode, it returns a zero. |
| [11] | power down | This bit always returns a zero because the ASIC does not implement a low-power mode. |
| [12] | auto-negotiation enable | Auto-negotiation can be disabled by one of two methods: hardware or software control.<br><br>If the ANEN input pin is driven to a logic zero, auto-negotiation is disabled by hardware control. If the auto-negotiation enable bit is written with a value of zero, auto-negotiation is disabled by software control.<br><br>When auto-negotiation is disabled by software control, writing a one to this bit or resetting the chip enables auto-negotiation.<br><br>Writing to this bit has no effect when auto-negotiation is disabled by hardware control. When read, this bit returns the value most recently written to it, or a one if it has not been written to since the last chip reset. |
| [13] | forced speed selection | This bit, when set to one, forces 100BASE-X operation if auto-negotiation is disabled by software control. When set to zero, this bit forces 10BASE-T operation if auto-negotiation is disabled by software control.<br><br>This bit has no effect on the speed selection if auto-negotiation is enabled (both the auto-negotiation pin and bit are enabled) or disabled (auto-negotiation pin is pulled low) by hardware control.<br><br>When this bit is read, it returns the value of the software-controlled forced speed selection only. |

**Control Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [14] | loopback | This bit, when set to one, enables loopback mode. When set to zero, it indicates normal operation. The loopback mode can also be cleared by resetting the NIC.<br><br>When this bit is read, it returns a one when the chip is in software-controlled loopback mode; otherwise, it returns a zero. |
| [15] | reset | This bit, when set to one using an MII write operation, resets the PHY. Writing a zero to this bit has no effect.<br><br>This bit clears itself after the reset process is complete and does not need to be cleared using a second MII write.<br><br>Writes to other bits in the Control register have no effect until the reset process is completed, which requires approximately 1 μs. Because this bit is self-clearing, after a few cycles from a write operation, it returns a zero when read. |

(3 of 3)

## Link Partner Ability

The Link Partner Ability register returns the advertised abilities received from the link partner during auto-negotiation.

**Link Partner Ability Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | 0 | 0 | | | | | | | | | | | |

**Link Partner Ability Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0:4] | link partner selector field | These bits reflect the value of the link partner's selector field. These bits are cleared anytime auto-negotiation is restarted or when the NIC is reset. |
| [5] | LP advertise 10BASE-T | This bit, when set to one, indicates that the link partner is capable of performing 10BASE-T operation. This bit is cleared anytime auto-negotiation is restarted or when the NIC is reset. |
| [6] | LP advertise 10BASE-T FDX | This bit, when set to one, indicates that the link partner is capable of performing 10BASE-T full-duplex operation. This bit is cleared anytime auto-negotiation is restarted or when the NIC is reset. |
| [7] | LP advertise 100BASE-X | This bit, when set to one, indicates that the link partner is capable of performing 100BASE-X operation. This bit is cleared anytime auto-negotiation is restarted or when the NIC is reset. |
| [8] | LP advertise 100BASE-X FDX | This bit, when set to one, indicates that the link partner is capable of performing 100BASE-X full-duplex operation. This bit is cleared anytime auto-negotiation is restarted or when the NIC is reset. |
| [9] | LP advertise 100BASE-4 | This bit, when set to one, indicates that the link partner is capable of performing 100BASE-T4 operation. This bit is cleared anytime auto-negotiation is restarted or when the NIC is reset. |

(1 of 2)

**Link Partner Ability Bit Descriptions (continued)**

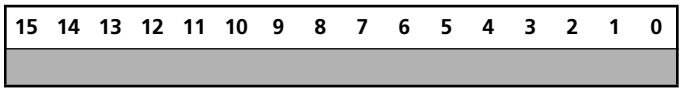
| Bit | Name | Description |
|---|---|---|
| [10] | LP pause operation | This bit, when set to one, indicates that the link partner is capable of pause operation. |
| [13] | LP remote fault | This bit returns a one when the link partner signals that a remote fault has occurred. The NIC copies the value to this register for the MAC layer to act upon. |
| [14] | LP acknowledge | This bit indicates that a device has successfully received the link partner's link code word. |

(2 of 2)

## PHYID High

The PHYID High register contains PHY identification data.

**PHYID High Register Format**



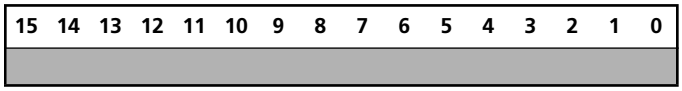

**PHYID High Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0:15] | Address 00010:ID MSBs | These bits are the organizationally unique identifier (OUI) most-significant bits.<br>Note that the two most-significant bits [23:22] of the OUI are not represented. |

## PHYID Low

The PHYID Low register contains PHY identification data.

**PHYID Low Register Format**




**PHYID Low Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0:15] | Address 00011:ID LSBs | These bits are the 16 least-significant bits of the PHY Identifier:<br>■ OUI [5:0] = These bits are the organizationally unique identifier (OUI) least-significant bits.<br>■ Model [5:0] = These bits are the vendor model number.<br>■ Revision [3:0] = These bits are the model revision number. |

**Status** The Status register contains various status and capabilities bits.

**Status Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | 0 | 0 | 0 | 0 | | | | | | | |

**Status Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | extended capability | The NIC supports extended capability registers and returns a one when this bit is read. |
| [1] | jabber detect | This bit returns a one if a jabber condition has been detected. After the bit is read once, or if the NIC is reset, it returns a zero.<br><br>This bit is implemented in 10BASE-T operation only. |
| [2] | link status | This bit returns a one when the link state machine is in link pass state, indicating that a valid link has been established. If a link has not been established, this bit returns a zero.<br><br>When a link failure occurs after the link pass state has been entered, the link status bit latches at zero and remains so until the bit is read. After the bit is read, it becomes a one when the link pass state is entered again. |
| [3] | auto-negotiation capability | This bit returns a one when read, regardless of whether or not the auto-negotiation function has been disabled. |
| [4] | remote fault | This bit returns a one if the link partner has detected a remote fault condition. When a remote fault occurs, the bit latches at one and remains at one until the register is read and the remote fault condition has been cleared.<br><br>This bit returns a zero if a remote fault condition has not been detected. |
| [5] | auto-negotiation complete | This bit returns a one if the auto-negotiation process is complete and the contents of the Auto-Negotiation Advertise, Link Partner Ability, and Auto-Negotiation Expansion registers are valid.<br><br>The bit returns a zero if the auto-negotiation process is not complete. |
| [6] | MF preamble suppression | This bit, when set to one, allows subsequent MII management frames to be accepted with or without the standard preamble pattern.<br><br>A one-time, two-cycle delay is required after this bit is set before the start of the next MII management frame (ST field) can begin. No added delay is required between frames in any future transmissions.<br><br>When this bit is set to zero, a preamble is always required. |
| [11] | 10BASE-T capability | This bit returns a one when read, indicating that the NIC is capable of 10BASE-T half-duplex operation. |
| [12] | 10BASE-T FDX capability | This bit returns a one when read, indicating that the NIC is capable of 10BASE-T full-duplex operation. |
| [13] | 100BASE-TX capability | This bit returns a one when read, indicating that the NIC is capable of 100BASE-TX half-duplex operation. |
| [14] | 100BASE-TX FDX capability | This bit returns a one when read, indicating that the NIC is capable of 100BASE-TX full-duplex operation. |

### TX Equalizer Coefficient Control

**TX Equalizer Coefficient Control Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | | | 0 | 0 | | | 0 | | 0 | 0 |

**TX Equalizer Coefficient Control Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [2] | equalizer init | Vendor-specific bit |
| [4] | freeze DFE | Vendor-specific bit |
| [5] | freeze FFE | Vendor-specific bit |
| [8:11] | coeffiecient select | Vendor-specific bit |

### TX Equalizer Coefficient Read/Write

**TX Equalizer Coefficient Read/Write Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | | | | | | | | | | | | | | |

**TX Equalizer Coefficient Read/Write Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0:13] | equalizer coefficient | Vendor-specific bit |

## 40-0483-00x ASIC Auto-Negotiation Registers

On 3C90xB NICs that use the 40-0483-00x ASIC, an integrated 802.3u auto-negotiation function handles auto-negotiation for 10BASE-T and 100BASE-TX media types. 100BASE-T4 is not implemented on the NICs.

The auto-negotiation component includes registers that can be used by driver software to read and change the auto-negotiation operational mode. These registers are accessed through the MII management interface. Although the auto-negotiation registers are accessed through the MII management interface, the auto-negotiation component does not use the MII data interface.

A physical layer device connected to the NIC's MII port may have its own auto-negotiation capability that is unrelated to the NIC's auto-negotiation capability. Software attempting to determine the link status of the NIC must consider the NIC's auto-negotiation capability as well as any external device's auto-negotiation capability.

Table 28 summarizes the names, addresses, and functions of the 40-0483-00x ASIC auto-negotiation and MII registers. The registers are described in alphabetical order in the section following the table.

**Table 28** Summary of 40-0483-00x ASIC Auto-Negotiation and MII Registers

| Address | Register Name | Description |
|---|---|---|
| 00 | MR0 Control | Contains control bits to reset, restart, and configure auto-negotiation. |
| 01 | MR1 Status | Contains various status and capabilities bits. |
| 02 | MR2 PHY Identifier 1 | Contains PHY identification data. |

(1 of 2)

**Table 28** Summary of 40-0483-00x ASIC Auto-Negotiation and MII Registers (continued)

| Address | Register Name | Description |
|---|---|---|
| 03 | MR3 PHY Identifier 2 | Contains PHY identification data. |
| 04 | MR4 Autonegotiation Advertisement | Controls which capabilities the NIC is allowed to advertise to the link partner. |
| 05 | MR5 Autonegotiation Link Partner Ability | Returns the advertised abilities received from the link partner during auto-negotiation. |
| 06 | MR6 Autonegotiation Expansion | Contains bits pertaining to expanded auto-negotiation functions. |
| 07 | MR7 Next Page Transmit | Contains bits that control the next page function, which is activated to allow the exchange of arbitrary pieces of data. |
| 08–15 | MR8-15 Reserved | Reserved. |
| 28 | MR28 Device-specific 1 | See the register bit definitions for a description of this register. |
| 29 | MR29 Device-specific 2 | See the register bit definitions for a description of this register. |
| 30 | MR30 Device-specific 3 | See the register bit definitions for a description of this register. |

(2 of 2)

## MR0 Control

The MR0 Control register contains bits to reset, restart, and configure auto-negotiation.

**MR0 Control Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MR0 Control Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [7] | collision test | When this bit is set to a logic zero, the ASIC asserts the COL signal in response to TXEN. This bit should only be set in loopback mode. |
| [8] | duplex mode | This bit returns one when the NIC is in full-duplex operation. It returns zero when the NIC is in half-duplex operation.<br>This bit is ignored when the autonegotiation enable bit (12) of this register is enabled. The default state is a logic zero.<br>This bit is ORed with the TXLED[A] and H_DUPLEX[D] signals during power-up and reset. |
| [9] | restart autonegotiation | This bit, when set to logic one, restarts the auto-negotiation process.<br>The autonegotiation complete bit in the Status Register is reset when this bit goes high. This bit is self-cleared when auto-negotiation starts, and its default is zero. |
| [10] | isolate | This bit, when set to one, indicates that MII outputs should be in high impedance state. The default state is a logic zero. |
| [11] | powerdown | This bit, when set to a logic one, places the NIC in a low-power state. While in powerdown state, the NIC responds to management transactions. The default is zero. |

(1 of 2)

**MR0 Control Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [12] | autonegotiation enable | This bit, when set to a logic one, enables the auto-negotiation process. The default is zero. |
| [13] | speed selection | This bit returns one when the current speed of operation is 100 Mbps. It returns zero when the current speed of operation is 10 Mbps.<br><br>This bit only affects operating speed when the auto-negotiation enable bit (12) in this register is disabled. This bit is ignored when the auto-negotiation enable bit is enabled. This bit is ANDed with the SPEEDLED[D] signal during power-up and reset. |
| [14] | loopback | When set, no data transmission takes place on the media and any received data is ignored. The loopback signal path contains all circuitry up to, but not including, the PECL I/O. The default is zero. |
| [15] | reset | This bit, when set, resets all registers to their default values. This bit is self-clearing. The default is zero. |

(2 of 2)

## MR1 Status

The MR1 Status register contains various status and capabilities bits.

**MR1 Status Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 0 | 0 | 0 | 0 | | | | | | | |

**MR1 Status Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | extended capability | This bit is always set to indicate that the NIC supports the extended register set (MR2 PHY Identifier 1 and beyond). |
| [1] | jabber detect | This bit is set whenever a jabber condition is detected. It remains set until it is read and the jabber condition no longer exists. |
| [2] | link status | This bit, when set to a logic one, indicates that a valid link has been established.<br><br>This bit has a latching function; a link failure causes the bit to clear and remain clear until it has been read through the management interface. |
| [3] | autonegotiation ability | This bit is always set to indicate the ability to perform auto-negotiation. |
| [4] | remote fault | This bit, when set, indicates that a remote fault has been detected. This bit remains set until it is cleared by a read. The default is a logic zero. |
| [5] | autonegotiation complete | This bit, when set to a logic one, indicates that the auto-negotiation process is complete and the contents of the MR4 Auto-Negotiation Advertisement, MR5 Auto-Negotiation Link Partner Ability, and MR6 Auto-Negotiation Expansion registers are now valid.<br><br>This bit is reset when auto-negotiation starts. The default is a logic zero. |
| [6] | suppress preamble | This bit is always set to indicate that the NIC accepts management frames with the preamble suppressed. |

(1 of 2)

**MR1 Status Bit Descriptions (continued)**

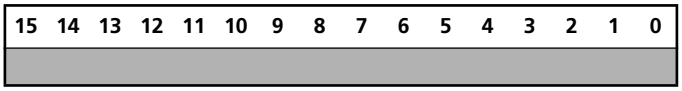
| Bit | Name | Description |
|---|---|---|
| [11] | 10BASE-T half-duplex ability | This bit is always set to a logic one, indicating 10BASE-T half-duplex ability. |
| [12] | 10BASE-T full-duplex ability | This bit is always set to a logic one, indicating 10BASE-T full-duplex ability. |
| [13] | 100BASE-TX half-duplex ability | This bit is always set to a logic one, indicating 100BASE-TX half-duplex ability. |
| [14] | 100BASE-TX full-duplex ability | This bit is always set to a logic one, indicating 100BASE-TX full-duplex ability. |
| [15] | 100BASE-T4 ability | This bit is always set to a logic zero, indicating that 100BASE-T4 is not supported. |

(2 of 2)

## MR2 PHY Identification

The MR2 PHY Identification register contains PHY identification data.

**MR2 PHY Identification Register Format**



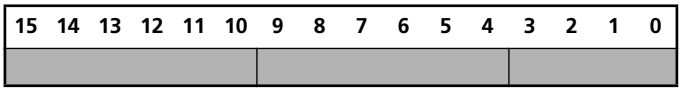

**MR2 PHY Identification Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0:15] | organizationally unique identifier (OUI) | Bits [3:24] of the OUI assigned to the PHY manufacturer by the IEEE are placed in bits [0:15] in this register and bits [10:15] of the MR3 PHY Identification register. |

## MR3 PHY Identification

The MR3 PHY Identification register contains PHY identification data.

**MR3 PHY Identification Register Format**



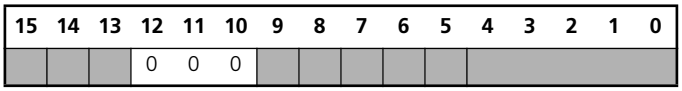

**MR3 PHY Identification Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0:3] | revision number | These bits indicate the present revision number (01h for the first revision). |
| [4:9] | model number | These bits indicate the 6-bit model number of the device. (The model number is 24h.) |
| [10:15] | organizationally unique identifier (OUI) | Bits [3:24] of the OUI assigned to the PHY manufacturer by the IEEE are to be placed in bits [0:15] in the MR2 PHY Identification register and bits [10:15] of this register. |

## MR4 Auto-Negotiation Advertisement

The MR4 Auto-negotiation Advertisement register controls which capabilities the NIC is allowed to advertise to the link partner.

**MR4 Auto-negotiation Advertisement Register Format**



**MR4 Auto-negotiation Advertisement Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0:4] | selector field | These read-only bits are hard-wired with the value 00001, as specified by the IEEE 802.3 specification. |
| [5] | 10BASE-T | When this bit is set, auto-negotiation advertises that the NIC is capable of 10BASE-T operation. |
| [6] | 10BASE-T full-duplex | When this bit is set, auto-negotiation advertises that the NIC is capable of 10BASE-T full-duplex operation. |
| [7] | 100BASE-TX | When this bit is set, auto-negotiation advertises that the NIC is capable of 100BASE-TX operation. |
| [8] | 100BASE-TX full-duplex | When this bit is set, auto-negotiation advertises that the NIC is capable of 100BASE-TX full-duplex operation. |
| [9] | 100BASE-T4 | This bit is always set to a logic zero. |
| [13] | remote fault | This bit, when set to a logic one, indicates a remote fault condition to the link partner. |
| [14] | acknowledge | This bit is the acknowledge bit from the link code word. |
| [15] | next page | This bit, when set, indicates that the next page function is activated to allow the exchange of arbitrary pieces of data. Data is carried by optional next pages of information. |

## MR5 Auto-Negotiation Link Partner Ability

The MR5 Auto-negotiation Link Partner Ability register returns the advertised abilities received from the link partner during auto-negotiation.

**MR5 Auto-negotiation Link Partner Ability Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |

**MR5 Auto-negotiation Link Partner Ability Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0:4] | selector field | This field contains the type of message sent by the link partner. For IEEE 802.3-compliant link partners, this field should read 00001. |
| [5:12] | technology ability field | This field contains the technology ability of the link partner. These bits are defined as shown in the MR4 Auto-Negotiation Advertisement register. |
| [13] | remote fault | This bit, when set, indicates that the link partner has a remote fault. |
| [14] | acknowledge | This bit, when set, indicates that the link partner has successfully received at least three consecutive and consistent fast link pulse (FLP) bursts. |
| [15] | next page | This bit, when set, indicates that the link partner wishes to engage in next page exchange. |

## MR6 Auto-Negotiation Expansion

The MR6 Auto-negotiation Expansion register contains bits pertaining to expanded auto-negotiation functions.

**MR6 Auto-negotiation Link Partner Ability Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | |

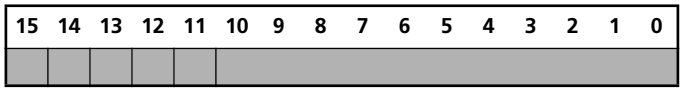
**MR6 Auto-negotiation Link Partner Ability Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | link partner auto-negotiation capable | This bit, when set, indicates that the link partner is capable of auto-negotiation. |
| [1] | page received | This bit is read-only and is latched high (R/LH). |
| [2] | next page able | This bit, when set, indicates that this device supports the next page function. |
| [3] | link partner next page able | This bit, when set, indicates that the link partner supports the next page function. |
| [4] | parallel detection fault | This bit, when set, indicates that a fault has been detected in the parallel detection function. This fault occurs when more than one technology detects concurrent link conditions. This bit can only be cleared by reading this register (R/LH). |

## MR7 Next Page Transmit

The MR7 Next Page Transmit register contains bits that control the next page function. The next page function allows the exchange of arbitrary pieces of data. (Data is carried by optional next pages of information.)

**MR7 Next Page Transmit Register Format**




**MR7 Next Page Transmit Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0:10] | message/unformatted code field | These bits are capable of 2,048 messages. Message code field definitions are described in Annex 28C of the IEEE 802.3u standard. |
| [11] | toggle | This read-only bit is used by the arbitration function to ensure synchronization with the link partner during next page exchange.<br>This bit always takes the opposite value of the toggle bit in the previously exchanged link code word. If the bit is a logic zero, the previous value of the transmitted link code word was a logic one. If the bit is a logic one, the previous value of the transmitted link code word was a logic zero.<br>The initial value of the toggle bit in the first next page transmitted is the inverse of the value of bit 11 in the base link code word and therefore may assume a value of logic one or logic zero. |
| [12] | acknowldege2 | This bit, when set to one, indicates that the device complies with the message. When set to zero, this bit indicates that the device cannot comply with the message. |
| [13] | message page | This bit, when set to one, indicates a formatted page. When set to zero, this bit indicates an unformatted page. This bit is used to differentiate a message page from an unformatted page. |
| [14] | acknowledge | This read-only bit is the acknowledge bit from the link code word. |
| [15] | next page | This bit, when set to one, indicates that there is an additional next page. When set to zero, this bit indicates that this is the last page. |

## MR28 Device-specific Register 1

**MR28: Device-specific Register 1 Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | |

**MR28 Device-specific Register 1 Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | link up 10 | This bit, when set to one, indicates that a 10 Mbps transceiver is up and operational. |
| [1] | link up 100 | This bit, when set to one, indicates that a 100 Mbps transceiver is up and operational. |
| [2] | force jam | This bit latches high until it is read (R/LH). |
| [3] | RX error status | This bit, when set to one, indicates a false carrier indication. This bit latches high until it is read (R/LH). |
| [4] | unlocked | This bit, when set to one, indicates that the transmit scrambler lost its synchronization. This bit latches high until it is read (R/LH). |
| [5] | disconnect | This bit, when set to one, indicates a disconnect. This bit is only valid in 10 Mbps mode. This bit latches high until it is read (R/LH). |
| [6] | autopolarity status | When the autopolarity function enable bit (3) in the MR30 Device Specific register is set, this bit returns a logic one, indicating that the NIC has detected and corrected a polarity reversal on the twisted-pair cable.<br>This bit is not valid in 100 Mbps operation. |
| [7] | code violation | This bit, when set to one, indicates that a Manchester code violation has occurred. The error code is output on the RXD lines. This is vendor-specific information.<br>This bit is only valid in 10 Mbps mode. This bit is latched high and clears itself only after it has been read or the device has been reset. |
| [8] | bad frame | This bit, when set, indicates that a packet has been received without a start frame delimiter (SFD).<br>This bit is only valid in 10 Mbps mode. This bit is latched high and clears itself only after it has been read or the device has been reset (R/LH). |

## MR29 Device-specific Register 2

**MR29 Device-specific Register 2 Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 0 | | | | 0 | | | | | | 0 | 0 |

**MR29 Device-specific Register 2 Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [2] | jam enable | This bit, when set, enables jam associated with carrier integrity to be ORed with collision. |
| [3] | carrier integrity enable | This bit, when set, enables carrier integrity. |
| [4] | scrambler/descrambler bypass | This bit, when set, disables scrambling/descrambling functions. |
| [5] | symbol aligner bypass | This bit, when set, disables the aligner function. |

(1 of 2)

**MR29 Device-specific Register 2 Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [6] | encoder/decoder bypass | This bit, when set, disables the 4B/5B encoder and 5B/4B decoder functions. |
| [8] | packet error indication enable | When this bit is set, a packet error code, which indicates that the scrambler is not locked, is reported on RXD[3:0] when RXER is asserted on the MII. A logic zero disables this function. |
| [9] | link error indication | When this bit is set, a link error code is reported on RXD[3:0] when RXER is asserted on the MII. The specific error codes are listed in the RXD signal description. A logic zero disables this function. |
| [10] | carrier sense select | This bit, when set to a logic one, asserts on receive only. If this bit is set to a logic zero, it asserts on receive or transmit. |
| [12] | 100 Mbps transmitter off | This bit, when set, forces TXOP low and TXON high. |
| [13] | generic reset 2 | This bit is vendor-specific. |
| [14] | generic reset 1 | This bit is vendor-specific. |
| [15] | management reset | This bit is the local management reset bit. When set, it causes the lower 16 registers and registers 28 and 29 to be reset to their default values. The bit is self-clearing. |

(2 of 2)

## MR30 Device-specific Register 3

**MR30 Device-specific Register 3 Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | |

**MR30 Device-specific Register 3 Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | no LP mode | This bit, when set, allows 10 Mbps operation with link pulses disabled.<br><br>If the NIC is configured for 10 Mbps operation, setting this bit does not affect operation. |
| [1] | serial select | This bit, when set, selects 10 Mbps serial mode operation.<br><br>If the NIC is in 100 Mbps mode, this bit is ignored. |
| [2] | reference select | When this bit is set, the external 10 MHz reference of signal REF10 is used for phase alignment. |
| [3] | autopolarity function enable | When this bit is set and the NIC is in 10 Mbps mode, the autopolarity function determines if the TOP link is wired with a polarity reversal.<br><br>If there is a polarity reversal, the NIC asserts the autopolarity status bit (6) of the MR28 Device-specific register and corrects the polarity reversal.<br><br>If this bit is a logic zero and the device is in 10 Mbps mode, the reversal is corrected. |
| [4] | extended line length enable | When this bit is set, the receive squelch levels are reduced, allowing reception of signals with a lower amplitude. This is valid in 10 Mbps mode only. |
| [5] | heartbeat enable | When this bit is set, the heartbeat function is enabled. This is valid in 10 Mbps mode only. |

# 12 OTHER REGISTERS

This chapter describes the following registers:

- BiosRomAddr
- BiosRomData
- DebugControl
- DebugData
- FifoDiagnostic
- Media
- NetworkDiagnostic
- PhysicalMgmt
- ResetOptions
- Timers and counters
- VlanEtherType

## BiosRomAddr

| | |
|---|---|
| Synopsis | Holds the address for direct I/O accesses of the BIOS ROM. |
| Type | Read/write |
| Size | 32 bits |
| Window | 0 |
| Offset | 4 |

The BiosRomAddr register applies to 3C90xB NICs only.

Together with the BiosRomData register, the BiosRomAddr register supports direct access to the BIOS ROM in I/O or memory space.

BiosRomAddr holds the address to be used for I/O accesses of the BIOS ROM through the BiosRomData port. To access a byte in the BIOS ROM, write the address of the byte to be accessed into BiosRomAddr. Then issue either a read or a write to the BiosRomData register. For reads, the ROM value will be returned by the read instruction. For writes, the new value will be programmed into the ROM upon completion of the write instruction.

**BiosRomAddr Register Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | | |

The Atmel PEROM devices supported by the NIC must be programmed in 64-byte pages. See the *Atmel Flash Memory Device Data Book* for information on programming PEROMs.

## BiosRomData

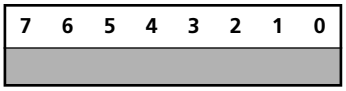
| | |
|---|---|
| Synopsis | Implements the data port for direct I/O accesses of the BIOS ROM. |
| Type | Read/write |
| Size | 8 bits |
| Window | 0 |
| Offset | 8 |

The BiosRomData register applies to 3C90xB NICs only.

Together with the BiosRomAddr register, the BiosRomData register supports direct access to the BIOS ROM in I/O or memory space.

BiosRomData is the data port for performing byte accesses of the BIOS ROM. A read of BiosRomData returns the ROM byte value from the location specified by the BiosRomAddr register. A write to BiosRomData causes the write data to be programmed into the ROM location specified by BiosRomAddr.

**BiosRomData Register Format**




The Atmel PEROM devices supported by the NIC must be programmed in 64-byte pages. See the *Atmel Flash Memory Device Data Book* for information on programming PEROMs.

## DebugControl

| | |
|---|---|
| Synopsis | Diagnostics control register. |
| Type | Read/write |
| Size | 16 bits |
| Offset | 74 |

The DebugControl register applies to 3C90xB NICs only.

DebugControl determines which sets of diagnostic data are visible in the DebugData register.

DebugControl is cleared by a reset.

**DebugControl Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | |

**DebugControl Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0:4] | debugSelect | This read/write field sets which set of data is visible in the DebugData register.<br>This field is ignored if the debugOverride bit is zero. |
| [15] | debugOverride | When this read/write bit is set, the value in the debugSelect field overrides the select input pins on the ASIC to determine what data set is output on the physical ASIC debug pins and visible in DebugData. |

## DebugData

| | |
|---|---|
| Synopsis | Diagnostics read-back register. |
| Type | Read-only |
| Size | 32 bits |
| Offset | 70 |

The DebugData register applies to 3C90xB NICs only.

DebugData is a 32-bit read-only diagnostic register for viewing the internal state of the ASIC. The debugSelect field in DebugControl determines which particular set of signals is visible in DebugData.

This register is intended for 3Com use only.

## FifoDiagnostic

| | |
|---|---|
| Synopsis | Provides diagnostic read access to the packet-buffering (FIFO) logic. |
| Type | Read/write |
| Size | 16 bits |
| Window | 4 |
| Offset | 4 |

**3C90x NIC FifoDiagnostic Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | | 0 | | | 0 | 0 | 0 | 0 | | | | | | |

The bits in the 3C90x FifoDiagnostic register provide various indications of transmit and receive FIFO failures.

Bits [5:0] control the built-in self-test (BIST) for the transmit and receive FIFO RAMs. These bits are used for IC-level testing only; driver software must always write zeros to these bits.

**3C90x NIC FifoDiagnostic Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | rxTestMode | This bit is a fast count for receive simulation. Software must not set this bit. |
| [1] | txTestMode | This bit is a fast count for transmit simulation. Software must not set this bit. |
| [2] | inMemBIST | This bit is the built-in self-test (BIST) mode. Write a one to this bit to start the BIST sequence. |
| [3] | inMemBFC | This bit is the BIST flag check. Set this bit to test for proper functioning of the inMemBF flag. |
| [4] | inMemBF | This bit is the BIST flag. When the NIC sets this flag, it indicates that a fault was detected in the internal SRAM. This bit is undefined until inMemBC becomes set. |
| [5] | inMemBC | This bit is the BIST complete. The NIC sets this bit when the BIST sequence is complete. |
| [10] | txOverrun | This bit is set when the transmit FIFO runs out of room to accept further frame data.<br>The assertion of this bit causes a hostError interrupt, which is set when a catastrophic error related to the bus interface occurs.<br>The TxReset or GlobalReset commands are used to recover from this condition. |
| [11] | rxOverrun | This bit is set when the receive FIFO is full. It is not necessary for frames to have been discarded for this bit to be set. However, frames received while this bit is set are discarded. This bit is informational only. |
| [13] | rxUnderrun | This bit, when set, causes a hostError interrupt that requires either an RxReset or GlobalReset command to clear.<br>An rxUnderrun occurs when the host reads data out of the receive FIFO faster than the network can fill it, or faster than the FIFO can supply data to the RxData register, resulting in the host accidentally reading invalid data. |
| [15] | receiving | This read-only bit is set whenever the NIC is receiving a frame into the receive FIFO. No particular action is expected on the part of the host based on the state of this bit. |

**3C90xB NICs FifoDiagnostic Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | | 0 | | 0 | | | | | | | | |

The bits in the 3C90xB NICs FifoDiagnostic register provide various indications of transmit and receive FIFO failures.

Bits [7:0] control the built-in self-test (BIST) for the transmit and receive FIFO RAMs. These bits are used for IC-level testing only; driver software must always write zeros to these bits.

**3C90xB NICs FifoDiagnostic Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | txBistEnable | This read/write bit enables the transmit FIFO BIST. |
| [1] | txBistControl | This read/write bit sets the transmit FIFO BIST mode. |
| [2] | txBistFlag | This read-only bit indicates the result of the transmit FIFO BIST. |
| [3] | txBistComplete | This read-only bit indicates that the transmit FIFO BIST is complete. |
| [4] | rxBistEnable | This read/write bit enables the receive FIFO BIST. |
| [5] | rxBistControl | This read/write bit sets the receive FIFO BIST mode. |
| [6] | rxBistFlag | This read-only bit indicates the result of the receive FIFO BIST. |
| [7] | rxBistComplete | This read-only bit indicates that the receive FIFO BIST is complete. |
| [9] | keepRxOverrun | This read/write bit determines how the NIC handles receive overrun packets. The default is zero, which causes the NIC to discard all overrun packets.<br>Setting this bit causes the NIC to keep and make visible all overrun packets that have been made visible to the host, so that they can be inspected for diagnostic purposes. |
| [11] | rxFull | This read-only bit is set when the receive FIFO is full.<br>This bit does not in itself indicate an overrun condition. However, if data is received while this bit is set, an overrun will occur.<br>This bit is informational only. This bit is cleared as soon as the receive FIFO is no longer full. |
| [15] | receiving | This read-only bit is set whenever the NIC is receiving a packet into the receive FIFO. No particular action is expected on the part of the host based on the state of this bit. |

## Media

### MacControl

| | |
|---|---|
| Synopsis | Allows control of parameters related to Media Access Control. |
| Type | Read/write |
| Size | 16 bits |
| Window | 3 |
| Offset | 6 |

The MacControl register provides for setting of MAC-specific parameters. It is cleared upon reset.

**MacControl Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | |

**MacControl Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | deferExtendEnable | Setting this bit enables the special deference mode, in which the time the transmitter defers after a transmission is extended to allow other stations collision-free access to the medium.<br><br>Clearing this bit causes the NIC to use standard IEEE 802.3 deference rules (except that values are scaled when the NIC is operating at 100 Mbps). |
| [4:1] | deferTimerSelect | This field is used to select the amount of time, in addition to the standard Inter-Frame Space (IFS) period, to defer when the NIC is operating in the special deference modes. The values and defer times are:<br>■ 0 = Standard IFS + 0 bit times<br>■ 1 = Standard IFS + 16 bit times<br>■ 2 = Standard IFS + 32 bit times<br>■ 3 = Standard IFS + 64 bit times<br>■ 4 = Standard IFS + 96 bit times<br>■ 5 = Standard IFS + 128 bit times<br>■ 6 = Standard IFS + 160 bit times<br>■ 7 = Standard IFS + 192 bit times<br>■ 8 = Standard IFS + 224 bit times<br>■ 9 = Standard IFS + 256 bit times<br>■ A = Standard IFS + 288 bit times<br>■ B = Standard IFS + 320 bit times<br>■ C = Standard IFS + 352 bit times<br>■ D = Standard IFS + 384 bit times<br>■ E = Standard IFS + 416 bit times<br>■ F = Standard IFS + 448 bit times<br><br>When the deferExtendEnable bit is clear, the special deference modes are disabled, and the value of deferTimerSelect is irrelevant. |
| [5] | fullDuplexEnable | Setting this bit configures the NIC to communicate with the hub or switch in a full-duplex manner. Specifically, it disables transmitter deference to receive traffic, allowing simultaneous receive and transmit traffic.<br><br>Setting this bit has the side effect of disabling CarrierLost statistics collection, because full-duplex operation requires carrier sense to be masked to the transmitter.<br><br>Software must issue TxReset and RxReset commands after changing the value of this bit.<br><br>For information on programming fullDuplexEnable, see "Setting the Duplex Mode" in Chapter 4. |

**MacControl Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [6] | allowLargePackets | This bit determines the packet size at which the oversizedFrame error is generated for receive packets.<br>The minimum packet sizes at which an oversizedFrame error will be flagged are:<br>■ allowLargePackets value 0 = 1519 minimum<br>■ allowLargePackets value 1 = 4495* minimum<br>*This value was calculated by taking the maximum FDDI frame size, 4500 bytes, and subtracting bytes for fields that have no Ethernet equivalent.<br>The packet size includes the destination and source addresses, the type/length field, and the FCS field. |
| [7] | extendAfterCollision | This bit applies to 3C90xB NICs only.<br>This bit determines the extended deference mode.<br>■ 0 = "Old-style" extended deference — Deference extension occurs after all transmissions, including collisions.<br>■ 1 = Deference extension occurs only after a collision, and only when this station was the last to transmit a packet. |
| [8] | flowControlEnable | This bit applies to 3C90xB NICs only.<br>Flow control enable.<br>■ 0 = Default. Treat all incoming packets normally.<br>■ 1 = Flow control enabled. Act upon incoming flow control (PAUSE) packets.<br>Note: This bit should not be set unless fullDuplexEnable is also set. |
| [9] | vltEnable | This bit applies to 3C90xB NICs only.<br>3Com-proprietary VLAN tagging (VLT) enable.<br>■ 0 = Default. Treat incoming packets as normal IEEE 802.3 frame format.<br>■ 1 = Enable VLT. Interpret the first four bytes of all incoming packets as the VLT field. Packets are received subject to the value set in the VlanMask register. |

(2 of 2)

## MediaOptions

| | |
|---|---|
| Synopsis | Provides access to the media options installed on the NIC. |
| Type | Read-only |
| Size | 16 bits |
| Window | 3 |
| Offset | 8 |

The MediaOptions register applies to 3C90xB NICs only.

The MediaOptions register shows what physical media connections are available in the NIC. This register is read in from EEPROM location 19 after a reset. It is cleared upon reset.

Some bits in the MediaOptions register represent media types that are not supported on 3C900B NICs and 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC. These are noted in the MediaOptions Bit Descriptions table.

**MediaOptions Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | | | | | | | |

**MediaOptions Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | baseT4available | This read-only bit, when set, indicates that a 100BASE-T4 PHY is available on the NIC through the MII.<br><br>This bit is not implemented on 3C900B NICs. It is programmed to zero. |
| [1] | baseTxAvailable | This read-only bit, when set, indicates that a 100BASE-TX PHY is available on the NIC, using the on-chip 100BASE-X interface.<br><br>If an MII-based 100BASE-TX PHY is available on the NIC (to be used instead of the on-chip 100BASE-X interface), this bit is zero, and the miiDevice bit in this register is set.<br><br>This bit is not implemented on 3C900B NICs. It is programmed to zero. |
| [2] | baseFxAvailable | This read-only bit, when set, indicates that a 100BASE-FX PHY is available on the NIC.<br><br>This bit is not implemented on 3C900B NICs. It is programmed to zero. |
| [3] | 10bTAvailable | This read-only bit, when set, indicates that a 10BASE-T encoder/decoder and transceiver are available on the NIC. |
| [4] | coaxAvailable | This read-only bit, when set, indicates that a 10BASE2 coaxial transceiver is available on the NIC.<br><br>This bit is not implemented on 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC. It is programmed to zero. |
| [5] | auiAvailable | This read-only bit, when set, indicates that a 10 Mbps AUI connector is available on the NIC.<br><br>This bit is not implemented on 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC. It is programmed to zero. |
| [6] | miiDevice | This read-only bit, when set, indicates that a PHY device is available through the MII. If the device is a 100BASE-T4 PHY, then the baseT4Available bit in this register is also set.<br><br>This bit is not implemented on 3C900B NICs. It is programmed to zero. |

(1 of 2)

**MediaOptions Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [8] | 10BaseFL | This bit applies to 3C90xB NICS with the 40-0502-00x ASIC only.<br>This read-only bit, when set, indicates that a 10BASE-FL transceiver is available on the NIC.<br>This bit is not implemented on 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC. It is programmed to zero. |

(2 of 2)

## MediaStatus

| | |
|---|---|
| Synopsis | Allows setting of media-specific parameters and provides media-specific status. |
| Type | Read/write |
| Size | 16 bits |
| Window | 4 |
| Offset | a |

The MediaStatus register provides for the setting of media-specific parameters, and for the reading of media-specific status indications.

**MediaStatus Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | | | | | 0 | | | | | | | 0 | 0 |

**MediaStatus Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [2] | crcStripDisable | This bit applies to 3C90xB NICs only.<br>The host asserts this bit if the receive packet's CRC is to be passed to the host as part of the data in the FIFO. The state of this bit does not affect the NIC's checking of the packet's CRC and its posting of CRC error status. This bit is cleared by a system reset.<br>To avoid confusing the FIFO logic, the value of this bit should only be changed when the receiver is disabled and the receive FIFO is empty. |
| [3] | enableSqeStats | This read/write bit must be set by the host to enable the SqeErrors statistics register to count SQE errors. This bit will normally only be set when an external transceiver across the AUI is used. |
| [4] | collisionDetect | This read-only bit provides a real-time indication of the state of the collisionDetect signal within the ASIC. |
| [5] | carrierSense | This read-only bit provides a real-time indication of the state of the carrierSense signal within the ASIC. |

(1 of 2)

**MediaStatus Bit Descriptions (continued)**

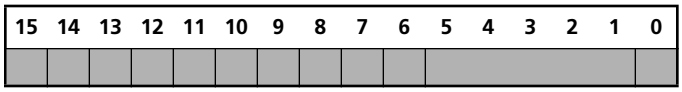
| Bit | Name | Description |
|---|---|---|
| [6] | jabberGuardEnable | This bit is for use only with the 10 Mbps twisted-pair transceiver.<br><br>When this read/write bit is set by the host, the NIC automatically shuts down transmissions if it detects that it is not sending transmission normally. This bit also enables the automatic reversal of polarity on the receive pair, if required. |
| [7] | linkBeatEnable | The host should set this read/write bit to require that the NIC detect the presence of the link beat to enable transmission. When this bit is false, the NIC is able to transmit packets with or without detecting link beat. |
| [9] | jabberDetect | This read-only bit is set whenever the NIC senses that it has been transmitting without interruption for much longer than the allowed transmit packet duration. When in this state, the NIC is disabled from further transmissions. The TxReset command is required to release the NIC from the jabber detect state. |
| [10] | polarityReversed | This read-only bit indicates that the twisted-pair transceiver has detected a reversal of polarity on its receive pair. If jabberGuardEnable is asserted, then the transceiver automatically corrects the polarity reversal. |
| [11] | linkDetect | This read-only bit provides a real-time indication of the twisted-pair transceiver link status for 10BASE-T, 100BASE-TX, and 100BASE-FX operation.<br><br>When the NIC is operating in 10BASE-T mode, this bit reflects the state of the link beat logic.<br><br>For 100BASE-TX or 100BASE-FX operation, this bit reflects the state of the link monitor process. For all of these modes, linkDetect is forced on whenever linkBeatEnable is cleared.<br><br>For MII operation, this bit is always set. |
| [12] | txInProg | This bit provides a real-time indication that a packet is being transmitted. This bit is used by drivers during underrun recovery to delay issuing a TxReset command. |
| [14] | dcConverterEnabled | This bit, when set, indicates that the 10BASE2 DC-DC converter has been enabled with the EnableDcConverter command.<br><br>This bit always reads as a zero on 3C90xB NICs that use the 40-0476-001 or 40-0483-00x ASIC. |
| [15] | auiDisable | This read-only bit is asserted whenever any media port except AUI has been selected.<br><br>This bit always reads as a zero on 3C90xB NICs that use the 40-0476-001 or 40-0483-00x ASIC. |

(2 of 2)

## NetworkDiagnostic

| | |
|---|---|
| Synopsis | Provides medium-dependent diagnostic access to the network interface logic, and a few other miscellaneous functions. |
| Type | Read/write |
| Size | 16 bits |
| Window | 4 |
| Offset | 6 |

**NetworkDiagnostic Register Format**




**NetworkDiagnostic Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | testLowVoltageDetector | Setting this bit tests the low-voltage detection circuit, which has the side effect of resetting the NIC. This bit always returns zero. |
| [5:1] | asicRevision | This field reflects the revision level of the ASIC.<br><br>The first operational silicon returns 00000b in this field. Significant revisions to the ASIC cause this field to be incremented.<br><br>For 3C90xB NICs, this field is equivalent to bits [4:0] in the RevisionId PCI configuration register. |
| [6] | upperBytesEnable | This read/write bit determines whether the upper bits of the BytesRcvdOk and BytesXmittedOk statistic registers are included in determining when an updateStatistics interrupt is generated.<br><br>■ For 3C90x NICs, when this bit is clear, as it is after a reset, the NIC defaults to a mode in which the updateStats bit in the IntStatus register is set whenever bit 15 of BytesRcvdOk or BytesXmittedOk becomes set.<br><br>■ For 3C90xB NICs, when this bit is clear, as it is after a reset, the chip defaults to a mode that is compatible with earlier-generation NICs in which updateStatistics is set whenever bit 15 of BytesRcvdOk or BytesXmittedOk becomes set.<br><br>When this bit is set, bit 15 of the BytesRcvdOk or BytesXmittedOk register and all four corresponding bits in the UpperBytesOk register must be set in order to cause an updateStatistics interrupt.<br><br>All drivers should set this bit and read UpperBytesOk after each read of the BytesRcvdOk and BytesXmittedOk registers. This reduces the number of updateStatistics interrupts to a more reasonable level (reducing the CPU utilization). |
| [7] | statisticsEnabled | This read-only bit indicates when the NIC is enabled to count the various statistical events. The value of this bit is affected by the StatisticsEnable and StatisticsDisable commands. |
| [8] | txFatalError | This bit is set if a jabber or txUnderrun occurs, indicating that the transmitter needs to be reset with the TxReset command. |

(1 of 2)

**NetworkDiagnostic Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [9] | transmitting | This bit is set whenever the NIC is transmitting or waiting to transmit (deferring). |
| [10] | rxEnabled | This read-only bit is set by the RxEnable command and is cleared by the RxDisable command, RxReset command, or a system reset. |
| [11] | txEnabled | This read-only bit is set by the TxEnable command and is cleared by the TxDisable command, TxReset command, or a system reset. |
| [12] | fifoLoopback | Setting this bit forces data loopback from the transmit FIFO directly into the receive FIFO.<br>When using FIFO loopback mode, it is the software's responsibility to ensure that the proper interpacket gap is inserted between packets, to avoid losing data in the receive path. To do this, the software must not load more than one transmit packet into the FIFO at a time. |
| [13] | macLoopback | Setting this bit causes the NIC to loop back transmissions at the output of the media access controller. |
| [14] | endecLoopback | ■ For 3C90x NICs and 3C90xB NICs with the 40-0502-00x ASIC, setting this bit causes the NIC to loop transmissions back to the receiver at the encoder/decoder for 10BASE-T, 10BASE2, and AUI operation. When either 100BASE-TX or 100BASE-FX is selected, the loopback path is within the PDT/PDR clock generation and recovery module.<br>■ For 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC, this is an external loopback and requires a loopback plug. |
| [15] | externalLoopback | Setting this bit enables reception of packets transmitted by the NIC. Address-filtering criteria must also be met for each packet received.<br>■ For 3C90x NICs and 3C90xB NICs with the 40-0502-00x ASIC in 100BASE-FX or 100BASE-TX operation, external loopback occurs inside the transceiver chip. For true "on-the-wire" loopback, use a loopback plug, clear all of the loopback bits, and set the fullDuplexEnable bit in the MacControl register.<br>■ For 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC, this is an external loopback and requires a loopback plug. |

(2 of 2)

The NetworkDiagnostic register provides diagnostic access to the network interface logic in the NIC.

The TxReset and RxReset commands must be issued after the value of any of the loopback bits is changed in the NetworkDiagnostic register or the value of the fullDuplexEnable bit changes in the MacControl register.

Table 29 summarizes the various loopback modes and the required values for the associated bits in the NetworkDiagnostic, MacControl, and PhysicalMgmt registers.

**Table 29** Loopback Modes

| Mode | fifoLoopback | macLoopback | endecLoopback | externalLoopback | fullDuplex Enable | cat5LinkTest Defeat |
|---|---|---|---|---|---|---|
| FIFO | 1 | 0 | 0 | 0 | x* | x |
| MAC | 0 | 1 | 0 | 0 | x | x |
| Encoder/decoder | 0 | 0 | 1 | 0 | x | † |
| "External" 100BASE-X‡ | 0 | 0 | 0 | 1 | x | 1 |
| True "on-wire" external 100BASE-X | 0 | 0 | 0 | 0 | 1 | 1 |
| External 10BASE-T** | 0 | 0 | 0 | 1 | x | x |
| External 10BASE2†† | 0 | 0 | 0 | 0 | 1 | x |
| External AUI‡‡ | 0 | 0 | 0 | 0 | 1 | x |
| External MII*** | 0 | 0 | 0 | 1 | x | 0 |

* x = don't care.
† 1 for 100BASE-TX/FX; x for others.
‡ Loopback through 100BASE-TX/FX transceiver chip—not a true "on-wire" loopback.
**Requires 10BASE-T loopback plug.
††Requires loopback plug or coax segment.
‡‡Loopback type determined by external AUI device.
***Loopback type controlled by MII device. May need to enable a loopback mode within MII device using the management interface.

## PhysicalMgmt

| | |
|---|---|
| Synopsis | Provides control over various physical layer functions. |
| Type | Read/write |
| Size | 16 bits |
| Window | 4 |
| Offset | 8 |

The PhysicalMgmt register contains control bits for the MII management interface, power management event generation, and 100BASE-X link test defeat.

**PhysicalMgmt Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |

Bits [2:0] control the MII management interface. The management interface is a two-wire serial interface connecting the NIC ASIC and any MII-compliant PHY devices residing on the NIC.

Driver software operates the management interface by writing and reading bit patterns that correspond to the physical waveforms required on the interface signals to this register. For more information on the management interface signal protocols, refer to the *Reconciliation Sublayer and Media Independent Interface* draft supplement to IEEE 802.3.

**PhysicalMgmt Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | mgmtClk | The MII management clock. This bit drives the management clock directly to the PMD devices. |
| [1] | mgmtData | The MII management data bit. When the mgmtDir bit in this register is set, the value written to this bit is driven onto the MDIO signal. When mgmtDir is cleared, data being driven by the PMD can be read from this bit. |
| [2] | mgmtDir | The MII data direction control bit. Setting this bit causes the ASIC to drive MDIO with the data bit written into mgmtData. |
| [15] | cat5LinkTestDefeat | Setting this bit defeats the link test function in the 100BASE-X reconciliation layer logic. This bit is for diagnostic purposes only; software should always write a zero to this bit.<br>This bit is implemented in 3C900B NICs and 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC but has no function. |

## ResetOptions

The ResetOptions register differs for the 3C90x and 3C90xB NICs.

### 3C90x NICs

| | |
|---|---|
| Synopsis | Provides read access to the power-on-reset options. |
| Type | Read/write |
| Size | 16 bits |
| Window | 3 |
| Offset | 8 |

The ResetOptions register for 3C90x NICs contains the power-on-reset (POR) bits, which are latched from certain ASIC pins upon hardware reset. Bits [8:0] make up the POR bits.

This register is also visible in the PCI configuration register space.

**3C90x NICs ResetOptions Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 0 | 0 | 0 | | | | | | | | | |

**3C90x NICs ResetOptions Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [0] | baseT4Available | This read-only bit, when set, indicates that a 100BASE-T4 PHY is available on the NIC through the MII. |
| [1] | baseTXAvailable | This read-only bit, when set, indicates that a 100BASE-TX PHY is available on the NIC. |
| [2] | baseFXAvailable | This read-only bit, when set, indicates that a 100BASE-FX PHY is available on the NIC. |
| [3] | 10bTAvailable | This read-only bit, when set, indicates that a 10BASE-T encoder/decoder and transceiver are available on the NIC. |

(1 of 2)

**3C90x NICs ResetOptions Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [4] | coaxAvailable | This read-only bit, when set, indicates that a 10BASE2 coaxial transceiver is available on the NIC. |
| [5] | auiAvailable | This read-only bit, when set, indicates that a 10 Mbps AUI connector is available on the NIC. |
| [6] | miiConnector | This read-only bit, when set, indicates that an MII-based connector is available on the NIC. |
| [8] | vcoConfig | This read/write bit determines whether the internal VCO or the external one is to be used, as defined below.<br>■ 0 = External VCO<br>■ 1 = Internal VCO |
| [12] | forcedConfig | This read/write bit, when set, places the ASIC in forced configuration mode. The NIC's PCI configuration register settings are ignored and the NIC is enabled with the following settings:<br>■ I/O base address = 200h<br>■ I/O target accesses = Enabled<br>■ Bus mastering = Enabled<br>■ Memory (BIOS ROM) accesses = Enabled |
| [15:13] | testMode | These read-only bits define the following special test modes:<br>■ 000 = 10 Mbps AUI thresholds<br>■ 001 = 10 Mbps receive<br>■ 010 = 10 Mbps transmit<br>■ 011 = 10BASE-T receive thresholds<br>■ 100 = 100BASE-TX/FX idle<br>■ 101–110 = Reserved<br>■ 111 = Normal operation |

(2 of 2)

## 3C90xB NICs

| | |
|---|---|
| Synopsis | Provides read access to various configuration and test mode bits. |
| Type | Read/write |
| Size | 16 bits |
| Window | 2 |
| Offset | c |

The ResetOptions register for 3C90xB NICs contains bits that indicate the configuration and test status of the NIC.

**3C90xB NICs ResetOptions Register Format**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | | | | | | | | | | | | | | |

**3C90xB NICs ResetOptions Bit Descriptions**

| Bit | Name | Description |
|---|---|---|
| [2:0] | featureSet | These read-only bits indicate the NIC's ability to support packet scheduling, extended deference, multicast filtering, and wake-up features.<br>■ 000 = Motherboard feature set: All advanced features are enabled except packet scheduling.<br>■ 001 = Low-cost NIC feature set: All advanced features are disabled.<br>■ 010 = Standard NIC feature set: All advanced features are enabled.<br>■ 100 = Server NIC feature set: All advanced features are enabled.<br>Other combinations are reserved. |
| [3] | d3ResetDisable | This bit indicates that the NIC is configured for operation in an AMD-style (pre-ACPI) Remote Wake-Up environment. Specifically, when the NIC is in the D3 power state, assertion of PCI RST# does not reset the NIC. |
| [4] | disableAdvFD | When this read/write bit is set, the auto-negotiation function is prevented from advertising full-duplex capability to its link partner.<br>This is similar to the baseTx and baseTxFullDuplex bits in the AutoNegAdvert register, except that this bit prevents full-duplex advertisement from reset time, instead of only from software load time.<br>For the NIC to be able to advertise full-duplex capability, this bit must be cleared and the appropriate bits in the AutoNegAdvert register must be set.<br>This bit is not implemented in 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC. It always returns zero. |
| [5] | disableAdv100 | When this read/write bit is set, the auto-negotiation function is prevented from advertising 100 Mbps capability to its link partner.<br>This is similar to the baseTx and baseTxFullDuplex bits in the AutoNegAdvert register, except that this bit prevents 100 Mbps advertisement from reset time, instead of only from software load time.<br>For the NIC to be able to advertise 100 Mbps capability, this bit must be cleared and the appropriate bits in the AutoNegAdvert register must be set.<br>This bit is not implemented in 3C900B NICs and 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC. It always returns zero. |
| [6] | disableAutoNeg | This read/write bit, when set, prevents the auto-negotiation function from starting the auto-negotiation process. Auto-negotiation still responds to accesses of its registers (MII-based), including setting the operating speed and duplex mode.<br>This bit may be cleared by software to allow auto-negotiation to proceed (this also requires that some of the auto-negotiation registers be written).<br>This bit is not implemented in 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC. It always returns zero. |

**3C90xB NICs ResetOptions Bit Descriptions (continued)**

| Bit | Name | Description |
|---|---|---|
| [7] | debugMode | This read/write bit is clear during normal operation. When set, it indicates that the ASIC's debug visibility mode is in effect. |
| [8] | fastAutoNeg | This read-only bit is used for simulation only. When set, it indicates a special auto-negotiation speed-up mode to decrease simulation time. |
| [9] | fastEE | This read-only bit is used for simulation only. When set, it indicates a special EEPROM speed-up mode to decrease simulation time. |
| [10] | forcedConfig | This read/write bit, when set, places the NIC into forced configuration mode. |
| [11] | testPdtPdr | When this read-only bit is set, the ASIC is in PDT/PDR test mode.<br>This bit is not implemented in 3C900B NICs and 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC. It always returns zero. |
| [12] | test100Tx | When this read-only bit is set, the ASIC is in a PDT bypass test mode.<br>This bit is not implemented in 3C900B NICs and 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC. It always returns zero. |
| [13] | test100Rx | When this read-only bit is set, the ASIC is in PDR bypass test mode.<br>This bit is not implemented in 3C900B and 3C90xB NICs with the 40-0476-001 or 40-0483-00x ASIC. It always returns zero. |

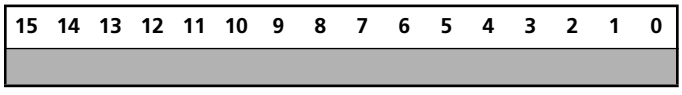
(2 of 2)

## Timers and Counters

### Countdown

| | |
|---|---|
| Synopsis | Provides a mechanism for the host to cause an interrupt to be generated by the NIC in a programmable time period. |
| Type | Read/write |
| Size | 16 bits |
| Offset | 36 |

**Countdown Register Format**




The Countdown register is a programmable down-counter that can cause the NIC to generate an interrupt when the counter expires.

Countdown has two modes of operation, which are selected by the countdownMode bit in the DmaCtrl register:

- When countdownMode is zero, Countdown is loaded by the host software with an initial countdown value. Thereafter, it decrements at a rate determined by the counterSpeed bit in DmaCtrl. When counterSpeed is clear, the count rate is once every 3.2 µs. When counterSpeed is set, the count rate is once every 320 ns. When Countdown reaches zero, it continues to count, wrapping to ffffh.
- When countdownMode is one, Countdown begins counting only when the dnComplete bit in the IntStatus register becomes set.

Countdown can cause an intRequested interrupt when it counts through zero. The interrupt is generated if the armCountdown bit in DmaCtrl is set at the time of the one-to-zero transition.

The armCountdown bit is managed solely by the hardware according to the following rules:

- Set when a nonzero value is written to Countdown
- Cleared when the value zero is written to Countdown, or when Countdown counts through zero

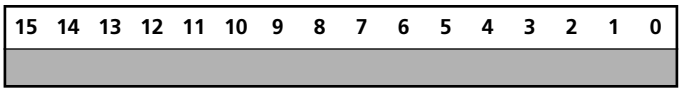
This means that when the host writes a nonzero value to Countdown, an interrupt is generated in a corresponding amount of time. By writing a zero value to Countdown, the host can suppress interrupts.

Countdown is cleared by reset.

## FreeTimer

| | |
|---|---|
| Synopsis | Provides a free-running counter for general timing purposes. |
| Type | Read-only |
| Size | 16 bits |
| Offset | 34 |

**FreeTimer Register Format**




The FreeTimer register is a free-running, read-only counter that increments at precise time intervals so that it can used for timing measurements. The count interval for FreeTimer is determined by the counterSpeed bit in the DmaCtrl register.

When counterSpeed is cleared, the count rate is once every 3.2 µs (four byte times at 10 Mbps). This yields a maximum measurable time interval of 200 ms. When counterSpeed is set, the count rate is once every 320 ns (four byte times at 100 Mbps), giving a maximum measurable time interval of 20 ms.

FreeTimer is cleared by hardware reset or the GlobalReset command.

### RealTimeCnt

| | |
|---|---|
| Synopsis | Provides a real-time counter for download scheduling. |
| Type | Read-only |
| Size | 32 bits |
| Offset | 40 |

**RealTimeCnt Register Format**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | |

The RealTimeCnt register applies to 3C90xB NICs only.

The RealTimeCnt register is a real-time counter that supports the packet download scheduling function.

RealTimeCnt counts continuously, incrementing every 800 ns (0.8 µs) and wrapping to zero when it reaches its maximum value. When a transmit packet is scheduled for download, the download starts when this register is greater than or equal to the value in the DPD's scheduleTime field.

RealTimeCnt is loaded with the value in the scheduleTime bit in the Schedule Time DPD entry when the loadTimeCnt bit is set. This has the side effect of causing the packet to be downloaded immediately.

RealTimeCnt is cleared by reset.

### Timer

| | |
|---|---|
| Synopsis | Provides a general-purpose timer function. |
| Type | Read-only |
| Size | 8 bits |
| Offset | 1a (3C90x NICs: also located at window 1, offset a) |

The Timer register contains an 8-bit counter that begins counting from zero upon the assertion of the interrupt signal. The host can use this function to make interrupt latency measurements. The counter increments by one every 3.2 µs. When the counter reaches ffh, it halts. This yields a maximum measurable interrupt latency of 816 µs.

When Timer is used to measure interrupt latency, it is suggested that Timer be read as late as possible in the interrupt service routine (just before dispatching to handle the interrupt reasons flagged in the IntStatus register) in order to include the fixed overhead of the interrupt handler itself.

To use Timer for general-purpose measurements at driver initialization time, ensure that the interruptLatch bit in the IntStatus register is clear (a pending interrupt would prevent the counter from starting), disable system interrupts, and issue a RequestInterrupt command to start the timer.

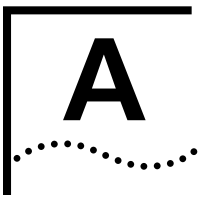
## VlanEtherType

| | |
|---|---|
| Synopsis | Supplies the EtherType value used to identify IEEE 802.1Q VLAN packets. |
| Type | Read/write |
| Size | 16 bits |
| Window | 7 |
| Offset | 4 |

The VlanEtherType register applies to 3C90xB NICs only.

The value in the VlanEtherType register allows the TCP/IP checksumming hardware to properly identify IEEE 802.1Q packets and exclude their VLAN information bytes from the checksum calculation.

On transmit and receive packets, the checksumming hardware compares the thirteenth and fourteenth bytes with the value in VlanEtherType. A match causes the hardware to skip over the thirteenth through sixteenth bytes of the packet.



# AUTOSELECT PSEUDO CODE

This appendix describes an AutoSelect sequence psuedo code.

## AutoSelect Sequence

The following psuedo code shows how to implement the NIC autoselect sequence, as desribed in Chapter 4.

```
// Definitions

// xcvrSelect        The xcvrSelect field in InternalConfig.
// mgmtData          The MII mgmt interface data read/write bit in PhysicalMgmt.
// mgmtClk           The MII mgmt interface clock bit in PhysicalMgmt.
// autoNegCapable    Bit 3 in the MII Status register of an MII device.
// autoNegComplete   Bit 5 in the MII Status register of an MII device.
// restartAutoNeg    Bit 9 in the MII Control register of an MII device.
// reset             Bit 15 in the MII Control register of an MII device.
// AutoNegAdvert     The MII register in an auto-negotiation-capable PHY that
//                   indicates the link speed/full-duplex capabilities of the PHY.
// AutoNegAbility    The MII register in an auto-negotiation-capable PHY that
//                   indicates the link speed/full-duplex capabilities received from
//                   link partner.
// The next 6 bits are bits read from the MediaStatus register. They indicate the
// presence on the NIC of the various possible media ports.
// baseTXAvailable   Indicates a 100BASE-TX port is available on the NIC.
// 10bTAvailable     Indicates a 10BASE-T port is available on the NIC.
// miiDevice         Indicates an off-chip MII device is available on the NIC.
// baseFXAvailable   Indicates a 100BASE-FX port is available on the NIC.
// auiEnable         Indicates an AUI port is available on the NIC.
// coaxAvailable     Indicates a 10BASE2 port is available on the NIC.


/*************** The main AutoSelect sequence ***************/
AutoSelect ()

// AutoSelect returns the selected port, link speed, and duplex mode
// or FALSE, indicating that no active port was found.
       if baseTXAvailable or 10bTAvailable
                     set xcvrSelect to "Auto-Negotiation"
                     if TryMII successful // First test NIC's internal
                     // Auto-Neg function for an active 10BASE-T
                     // or 100BASE-TX link
                     return results from TryMII
       else if miiDevice
                     set xcvrSelect to "MII"
                     if TryMII successful // then test any MII device
                     return results from TryMII
       else if baseFXAvailable
                     set xcvrSelect to "100BASE-FX"
                     if TryLinkDetect successful
                     return 100BASE-FX, 100MBPS, HALF_DUPLEX
```

```
        else if auiEnable
                        set xcvrSelect to "AUI"
                        if TryLoopback(AUI) successful
                        return AUI, 10MBPS, HALF_DUPLEX
        else if coaxAvailable
                        set xcvrSelect to "10BASE-2"
                        if TryLoopback(10BASE-2) successful
                        return 10BASE-2, 10MBPS, HALF_DUPLEX
        else return FALSE// no active port found

/*************** Sub-routines ***************/
TryLinkDetect()                         // returns TRUE when good link
        download self-directed packet   // this unpartitions 3Com on certain hubs
        program RxFilter for Promiscuous operation
        issue TxEnable and RxEnable
        for 1 to 65535
                        read RxStatus for any received packets
                                        if packet(s) received without error, return TRUE
                                        if packet(s) received with error, discard
                        read linkDetect
                                        if off, return FALSE
                        check carrierSense bit and accumulate result
        // fell out of the loop, so no good packets in 65535 tries
        if carrierSense on > 25% of time, return FALSE // all those carrierSenses
                                                       // should have yielded a good packet
        if carrierSense on < 25% return TRUE           // assume the link is good, 25% is
                                                       // fairly arbitrary
TryMII ()

// TryMII checks the on-chip auto-negotiation logic or an off-chip MII PHY,
// depending upon what is set in xcvrSelect by the caller.
// It exits when it finds the first device with a good link. TryMII returns the
// selected port, link speed, and duplex mode, or FALSE if no good link found

        if xcvrSelect is set to "Auto-Negotiate"
                        if TryPHY(11000b) successful // the on-chip auto-neg logic
                        return AUTO-NEG, link speed, and duplex mode
        else // xcvrSelect is set to "MII"
                        make sure mgmtDir is clear
                        read mgmtData : 1 indicates a PHY present
                        if no PHY, return FALSE
                        else // continue, find all PHY devices attached
                        for PHYAD = 0 to 31 except 11000b
                                read MII Control register
                                if a PHY responds, store that PHYAD value
                        if no response to any PHYAD, return FALSE
                        for all responding PHYAD values
                                if TryPHY(PHYAD) successful
                                        return MII, link speed, and duplex mode
                        // fell out of loop with no successful PHYAD
                        return FALSE
TryPHY(PHYAD)
// TryPHY checks the auto-negotiation function in the PHY at PHYAD
// It can also be extended to include link detection for non-IEEE 802.3u
// auto-negotiation devices, for instance the BCM5000.
// TryPHY returns the link speed and duplex mode (caller knows which
// port is selected).

        issue PHY reset to device at PHYAD
        poll on reset bit until cleared
```

```
                if reset bit not cleared in 2 seconds return FALSE
        read MII Control register again // bits aren't latched - read again to make sure
        if reset bit set return FALSE  // didn't really reset
        read MII Status register, check that Extended registers supported
        if Extended supported          // means auto-negotiation might be supported
                read PHY ID registers// save these values to aid
                                // PHY-specific bug fixes
                if (autoNegCapable and !autoNegComplete)
                        restart Auto-Negotiation by setting restartAutoNeg
                        poll on autoNegComplete for up to 2 sec
                        if !autoNegComplete return FALSE // never finished, go to next PHY
                        // auto-neg completed, see what happened
                        read AutoNegAdvert and AutoNegAbility registers
                        negotiated link mode is the highest common bit
                                set in the range [5..9]
                        if a common bit found
                                return link speed and duplex mode
                                else return FALSE
        else // no Extended reg or no auto-neg support
                        (do PHY-specific tests, i.e., BCM5000 sequence)
                        return link speed and duplex mode or FALSE

TryLoopback(port)               // try a loopback packet: use for 10BASE2 or AUI port
        if (port == 10BASE-2) issue EnableDcConverter command
        for 1 to 3              // give a port 3 chances to complete a loopback
                if TestPacket successful
                        if (port == 10BASE-2) issue DisableDcConverter command
                        return TRUE
        if (port == 10BASE-2) issue DisableDcConverter command
        return FALSE

TestPacket()
        set fullDuplexEnable in MacControl register
        set RxFilter to enable Individual Address matches
        issue RxEnable and TxEnable
        setup a UPD for a receive packet
        download a self-directed packet
        poll on txComplete in TxStatus register
        reset transmitter
        poll on upComplete in UPD UpPktStatus field for up to 1.5 sec
        if packet complete and no error return TRUE
        else return FALSE
        clear fullDuplexEnable in MacControl register

QuietAdapter ()
        set xcvrSelect to 10BASE-T
        clear linkBeatEnable, enableSqeStats, and jabberGuardEnable in MediaStatus
        wait 1.5 seconds
        issue TxReset and RxReset
```

# B

# PROGRAMMING THE MII MANAGEMENT INTERFACE

The Media-Independent Interface (MII) management interface is used to access registers in an MII PHY device.

On 3C90xB NICs, the on-chip auto-negotiation registers appear as a PHY device and are accessible through the management interface. A 3C90xB NIC may also have an off-chip PHY device with registers visible across the management interface.

The internal PHY address is 18H.

Register accesses across the MII management interface occur serially. Drivers control access with the mgmtClk, mgmtData, and mgmtDir bits in the PhysicalMgmt register. The direction of the serial transmission is controlled by mgmtDir; it is set when bits are written to the PHY device, and cleared when bits are read from PHY. Data bits are read from and written to the mgmtData bit. The mgmtClk bit supplies the synchronization clock for the interface.

## Management Frame Formats

The serial bit sequences used to read and write registers are called frames. The following table shows the frame formats for register read and write accesses for the 3C90xB NICs. Each box defines the bits in a certain frame field. Each field consists of one or more read, write, or Z cycles. The fields are sent across the interface from left to right.

The Read and Write frame sequences for the 3C90xB NICs are described in the sections following the table.

**Table 30** Management Frame Formats

| Type | PRE* | ST | OP | PHYAD | REGAD | TA | DATA | IDLE |
|---|---|---|---|---|---|---|---|---|
| Read | 1...1 | 01 | 10 | AAAAA | RRRRR | Z0 | DDDDDDDD DDDDDDDD | Z |
| Write | 1...1 | 01 | 01 | AAAAA | RRRRR | 10 | DDDDDDDD DDDDDDDD | Z |

* This is 32 consecutive "1" bits.

### Read Frame

A read frame consists of the following sequence.

1. Set the mgmtDir bit in the PhysicalMgmt register.
2. Execute write cycles to transmit the bits in the first five read frame fields, one bit per cycle.
3. Execute a Z cycle to prepare the interface to receive read data bits.

4 Execute a single read cycle. The PHY should be driving a zero to indicate its intention to respond to the read access. A one indicates that no PHY is responding and the data to follow is invalid.

5 Execute 16 read cycles to read the data field. Data bits are received starting with register bit 15 and ending with register bit 0.

6 Execute a Z cycle to terminate the frame.

**Write Frame** A write frame consists of the following sequence.

1 Set the mgmtDir bit in the PhysicalMgmt register.

2 Execute write cycles to transmit the bits in the first six write frame fields, one bit per cycle.

For 3C90xB NICs with the 40-0483-00x ASIC, execute write cycles to transmit bits in the first seven write frame fields, one bit per cycle.

3 Execute 16 write cycles to transmit the bits in the data field. Data bits are transmitted starting with register bit 15 and ending with register bit 0.

4 Execute a Z cycle to terminate the frame.

**Read Cycle** To read a single MII data bit from the interface, follow this procedure.

1 Clear the mgmtClk bit in the PhysicalMgmt register.

2 Wait a minimum of 200 ns.

Back-to-back I/O cycles on the PCI bus generally guarantee this, but drivers may use an arbitrarily long timer to time clock transitions.

3 Set mgmtClk.

4 Wait a minimum of 200 ns.

5 Read the next data bit from the mgmtData bit.

6 Wait a minimum of 200 ns.

**Write Cycle** To write a single MII data bit to the interface, follow this procedure.

1 Clear the mgmtClk bit in the PhysicalMgmt register.

2 Wait a minimum of 200 ns.

Back-to-back I/O cycles on the PCI bus generally guarantee this, but drivers may use an arbitrarily long timer to time clock transitions.

3 Set mgmtClk.

4 Write the desired data bit to mgmtData.

5 Wait a minimum of 200 ns.

**Z Cycle** This procedure is used during the turnaround portion of a register read frame. It terminates transmission and resets the mgmtDir bit.

1 Clear the mgmtClk bit in the PhysicalMgmt register.

2 Wait a minimum of 200 ns.

3 Set mgmtClk.

4 Clear the mgmtDir bit.

5 Wait a minimum of 200 ns.

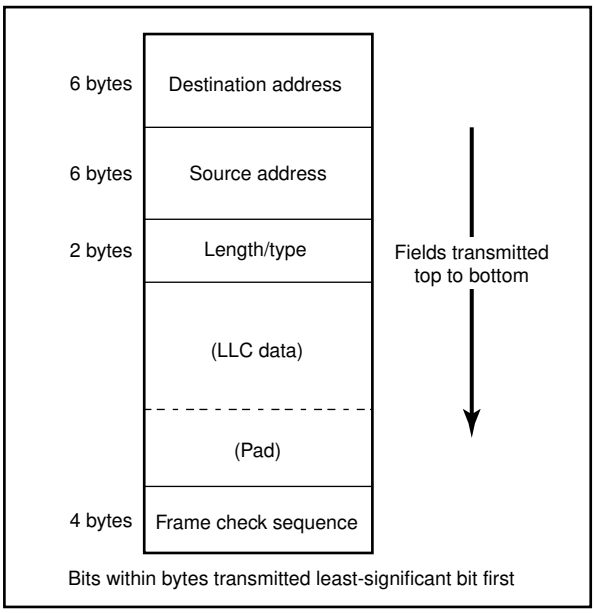
# C

# FRAME FORMATS

This appendix illustrates the frame formats.

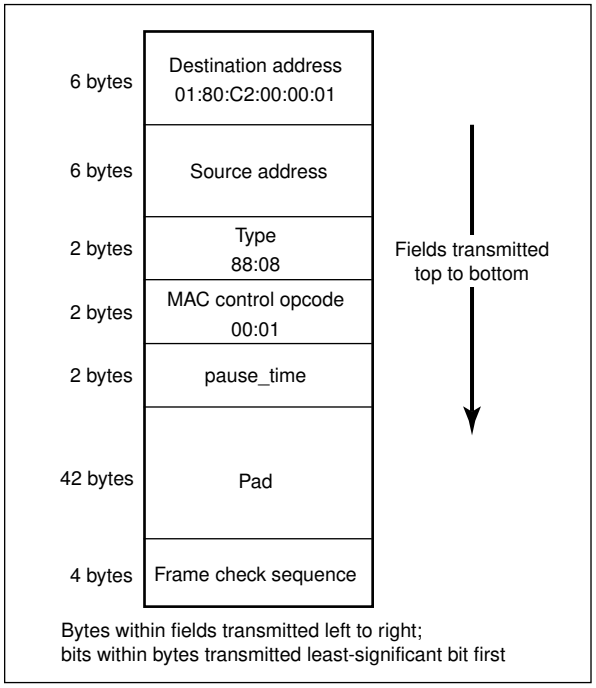
## IEEE 802.3 MAC Frame Format

**Figure 30** IEEE 802.3 MAC Frame Format



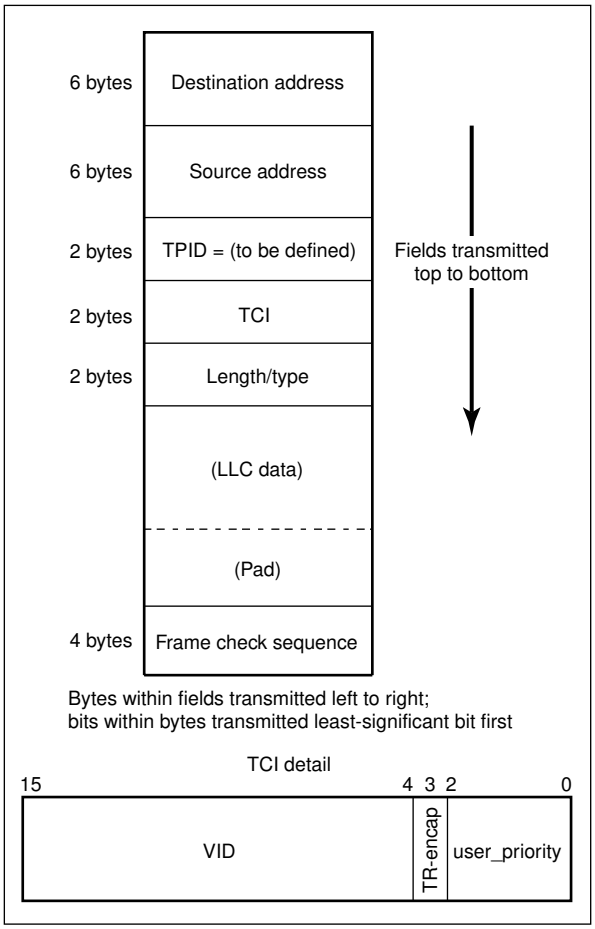
## IEEE 802.3x PAUSE Frame Format

**Figure 31** IEEE 802.3x PAUSE Frame Format



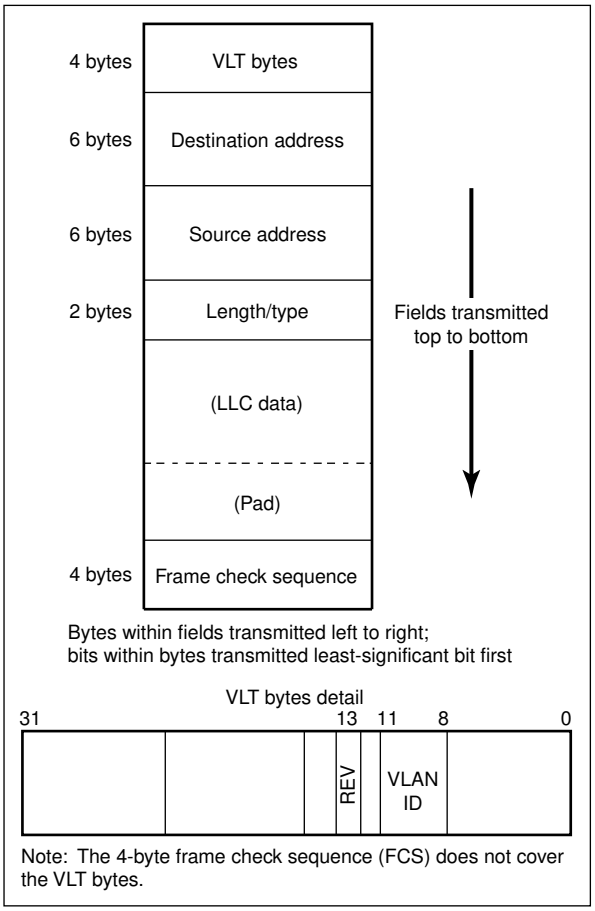
## IEEE 802.1q Frame Format

**Figure 32** IEEE 802.1q Frame Format



## 3Com VLT Frame Format

**Figure 33** 3Com VLT Frame Format



# D

# ERRATA LIST AND SOFTWARE SOLUTIONS

This appendix describes NIC anomalies and their software solutions.

## 3C90x NICs

Table 31 lists the anomalies with the 3C90x NICs.

**Table 31** 3C90x NICs (40-0336-00x ASIC) Anomalies

| ASIC(s) | Anomaly | Description | Resolution |
|---|---|---|---|
| 40-0336-00x | Rx collision discard race | An rxComplete interrupt is generated, but when the driver checks the NIC, there is no receive packet available and the receive FIFO is empty. | Software sets the RxEarlyThresh register to 60 bytes or greater, which causes the receive FIFO to wait until the slot time has expired before making the packet visible.<br>Because the receive FIFO logic never automatically discards a packet whose length is greater than the slot time, the problem is avoided. |
| 40-0336-00x | IBM predrive frame/CBE | 3C90x NICs predrive the following immediately after they see a GRANT and no other FRAME:<br>■ PCI<br>■ FRAME<br>■ AD signal<br>■ CBE signal<br>This causes problems on an IBM machine because of a bug in the arbiter where a FRAME was being driven even though there was no GRANT.<br>This problem has not been fixed in metal. | There is no software solution. |

## 3C90xB NICs

Table 32 lists the anomaly with the 3C90xB NICs that use the 40-0502-00x ASIC.

**Table 32** 3C90xB NICs (40-0502-00x ASIC) Anomaly

| ASIC(s) | Anomaly | Description | Resolution |
|---|---|---|---|
| 40-0502-00x | NIC hang caused by bus master receive uploads | Bus master receive data uploads may hang the NIC under certain circumstances.<br>When the hang is detected, all data appears to have been transferred correctly to system memory, but the bus master logic is stuck in one of two states.<br>A host reset clears the hang condition. | 3C90xB drivers can disable upload parallel tasking on bus master uploads by setting the upRxEarlyEnable bit in the PktStatus register, and setting the RxEarlyThresh register to a large value, forcing the packet upload to finish before bus mastering begins. |

Table 33 lists the anomalies with 3C90xB NICs that use the 40-0483-00x ASIC.

**Table 33** 3C90xB NICs (40-0483-00x ASIC) Anomalies

| ASIC(s) | Anomaly | Description | Resolution |
|---|---|---|---|
| 40-0483-001 | PowerManagementControl access problem | The PowerManagementControl register is a word-wide register in the PCI configuration space that must be accessed as a word only.<br>Byte accesses result in incorrect or indeterminate data. | Access the PowerManagementControl register with a word only. |
| 40-0483-001 | PCI bus turnaround | Upon releasing a target read cycle, the NIC continues to drive the AD lines through the PCI bus turnaround clock cycle. | |
| 40-0483-001, 40-0483-002 | Boot ROM and D3 state | Because of an implementation in the PCI Power Management specification, the NIC blocks PCI resets while in a D3 state, in order to maintain a sleep state. As a result, no memory or I/O cycles are permitted in the D3 state. Consequently, a boot ROM is not visible when the NIC is in a D3 state. | In the BIOS, change the NIC from a D3 state to a D0 state. |
| 40-0483-001, 40-0483-002 | Excessive CRC errors | A problem with the process-tuning block in the ASIC may cause the incoming data streams to be corrupted, resulting in excessive CRC errors and dropped packets. | Update the driver using *EtherDisk* diskette version 3.1 or later. This update disables the process-tuning block. |
| 40-0483-001 | 2 MHz Rx reject | If the Rx amplitude falls within a specific window (approximately 70 mV), the 10 Mbps receiver does not reject a 2 MHz input stream. | This is a parametric issue only. |
| 40-0483-001, 40-0483-002, 40-0483-004 | 100 Mbps Tx duty cycle | The Tx duty cycle is out of IEEE specifications by approximately 500 ps. This may result in an increased bit error rate. | |

# INDEX

## U



## V



## W



## Z

# INDEX OF REGISTERS

**T**



**U**



**V**

# INDEX OF BITS

## D



## E



## F

## U



## V



## W



## X