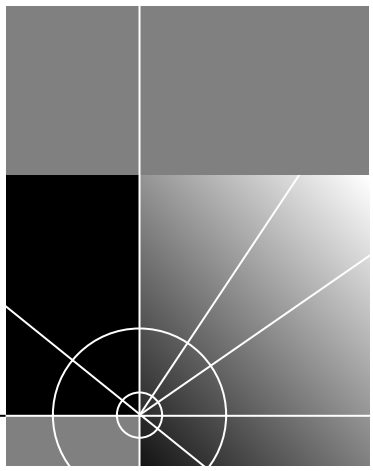
3C90xC NICs
Technical Reference

3Com® EtherLink® and EtherLink Server 10/100 PCI network interface cards



http://www.3com.com/

Part Number: 89-0931-000
Published September1999

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

3Com Corporation reserves the right to revise this documentation and to make changes in content from time to time without obligation on the part of 3Com Corporation to provide notification of such revision or change.

3Com Corporation provides this documentation without warranty of any kind, either implied or expressed, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. 3Com may make improvements or changes in the product(s) and/or the program(s) described in this documentation at any time.

UNITED STATES GOVERNMENT LEGENDS:
If you are a United States government agency, then this documentation and the software described herein are provided to you subject to the following restricted rights:

For units of the Department of Defense:
Restricted Rights Legend: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) for Restricted Rights in Technical Data and Computer Software Clause at 48 C.F.R. 52.227-7013. 3Com Corporation, 5400 Bayfront Plaza, Santa Clara, California 95052-8145.

For civilian agencies:
Restricted Rights Legend: Use, reproduction, or disclosure is subject to restrictions set forth in subparagraph (a) through (d) of the Commercial Computer Software – Restricted Rights Clause at 48 C.F.R. 52.227-19 and the limitations set forth in 3Com Corporation's standard commercial agreement for the software. Unpublished rights reserved under the copyright laws of the United States.

If there is any software on removable media described in this documentation, it is furnished under a license agreement included with the product as a separate document, in the hard copy documentation, or on the removable media in a directory file named LICENSE.TXT. If you are unable to locate a copy, please contact 3Com and a copy will be provided to you.

Unless otherwise indicated, 3Com registered trademarks are registered in the United States and may or may not be registered in other countries.

3Com and EtherLink are registered trademarks of 3Com Corporation. Lanworks is a trademark of 3Com Corporation.

Magic Packet is a trademark of Advanced Micro Devices, Inc. Atmel is a trademark of Atmel Corporation. Broadcom is a trademark of Broadcom Corporation. Dell is a registered trademark of Dell Computer Corporation. IBM is a registered trademark of International Business Machines Corporation. Lucent Technologies is a trademark of Lucent Technologies, Inc. National Semiconductor is a registered trademark of National Semiconductor Corporation.

Other brand and product names may be registered trademarks or trademarks of their respective holders.

CONTENTS

4 CONFIGURATION

5 EEPROM

FIGURES

TABLES

1 INTRODUCTION

This technical reference describes the basic architecture and defines the programming interface of the 3Com® EtherLink® and EtherLink Server 10/100 PCI network interface cards (NICs). The NIC models are listed in Table 1.

Table 1 3C90xC NICs

Model	Description	Notes
3C905C-TX	EtherLink 10/100 PCI NIC for Complete PC Management	Includes boot ROM socket
3C905C-TX-M	EtherLink 10/100 PCI NIC for Complete PC Management	Includes MBA boot flash ROM.
3C980C-TXM	EtherLink Server 10/100 PCI NIC	Includes MBA boot flash ROM.

Specifications in this technical reference apply to all listed NICs unless the text designates a specific model.

3C90xC NIC Features

The 3C90xC NIC contains the following features:

- 2 KB transmit FIFO and 2 KB receive FIFO.
- True dual-channel DMA engine.
- Enhanced scatter-gather engines reduce the number of I/O operations required to support data transfers (compared to the 3C905B NIC).
- 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 improved multicast packet handling (compared to the 3C905B NIC).
- Support for VLANs and IEEE 802.3x flow control functions.
- Support for IEEE 802.3u auto-negotiation (10BASE-T and 100BASE-TX).
- Support for ACPI Power Management.
- Support for wake-up events
- Improved bus master efficiency through use of optimal PCI memory commands and support of larger burst lengths.
- 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
- TCP/IP checksum features.

- Direct register access to BIOS ROM.
- An integrated 100 Mbps PHY that eliminates the need for an external 100 Mbps transceiver.
- Two-wire SMBus (System Management Bus) that provides register, configuration, and transmit and receive FIFO access.
- Flexible EEPROM loads; any internal register can be written, commands can be issued, and transmit FIFO can be loaded.
- Keep-alive packets that can be stored in the transmit FIFO and sent out at specific time intervals during sleep mode. These packets can also be "linked" to a specific pattern match effectively performing an "ack" function.
- SOS hardware pins that are linked to one of the keep-alive packets. These are intended for alerting the management console that some abnormal event (such as a case intrusion) has occurred.
- PCI 2.2 compliance, which includes proper handling of PCIReset, sensing 3.3V-AUX, a very low power mode (less than 20 mA), and reporting total current consumption.

About This Technical Reference

This technical reference contains programming interface information that software engineers, independent software developers, and test engineers require to write device drivers, diagnostic programs, and production test software for 3C90xC NICs.

This information includes:

- Theory of operation; for example, how transmission and reception occur.
- Register set, including the size, type, address, and function of each register and the functions of the bits in the register.

The information in this reference is language-independent. It applies regardless of the programming language you use to write the driver or other software program.

In this reference, addresses refer to physical addresses, not to logical or virtual addresses. Numeric values other than **Decimal** values are presented in the following formats:

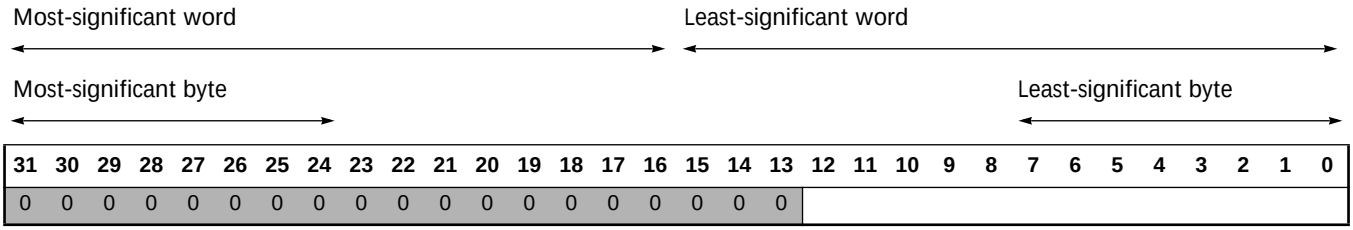
Format	Description	Example
#'rZZZZ	# is the number of bits.	6'b100101 is a 6-bit binary notation.
	' is a delimiter.	6'h25 is a 6-bit hexadecimal notation.
	r is the radix (b for binary and h for hexadecimal).	
	ZZZZ is the value.	
ZZZr	ZZZ is the value.	100101b is a binary notation.
	r is the radix (b for binary and h for hexadecimal).	25h is a hexadecimal notation.

Terms and Acronyms

The following terms and acronyms are used in this reference:

Term or Acronym	Meaning
ACK	Acknowledge.
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 NIC.
DPD	Download packet descriptor.
FIFO	First in, first out.
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.
ISR	Interrupt Service Routine
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.
SMBus	System Management Bus
UDP	User datagram protocol.
UPD	Upload packet descriptor.
Upload	The process of transferring receive data from 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 the bit map indicates the following information:

- Shaded areas indicate one of the following:
  - Read-only bits. These bits read back the default values shown. If no value is shown, the read-back value varies.
  - Unimplemented, reserved bits. These bits may be placeholders for possible use in a future revision of the NIC, or they may provide diagnostic information. Reserved bits are writable, but they do not control any function. They disregard data written to them and return zeros when they are read. To maintain compatibility with future versions of the NIC, drivers should write zeroes to reserved bits.
- Unshaded areas indicate active bits. The functions of these bits are described in the register descriptions. A value in an unshaded bit indicates that the driver must write that value to the bit.
- Vertical lines mark the boundaries of fields of bits (for example, [12:0]).

Default bit values are indicated as follows:

- 0 and 1 are known default states.
- x is a bit that is not initialized at reset; thus, its value varies.

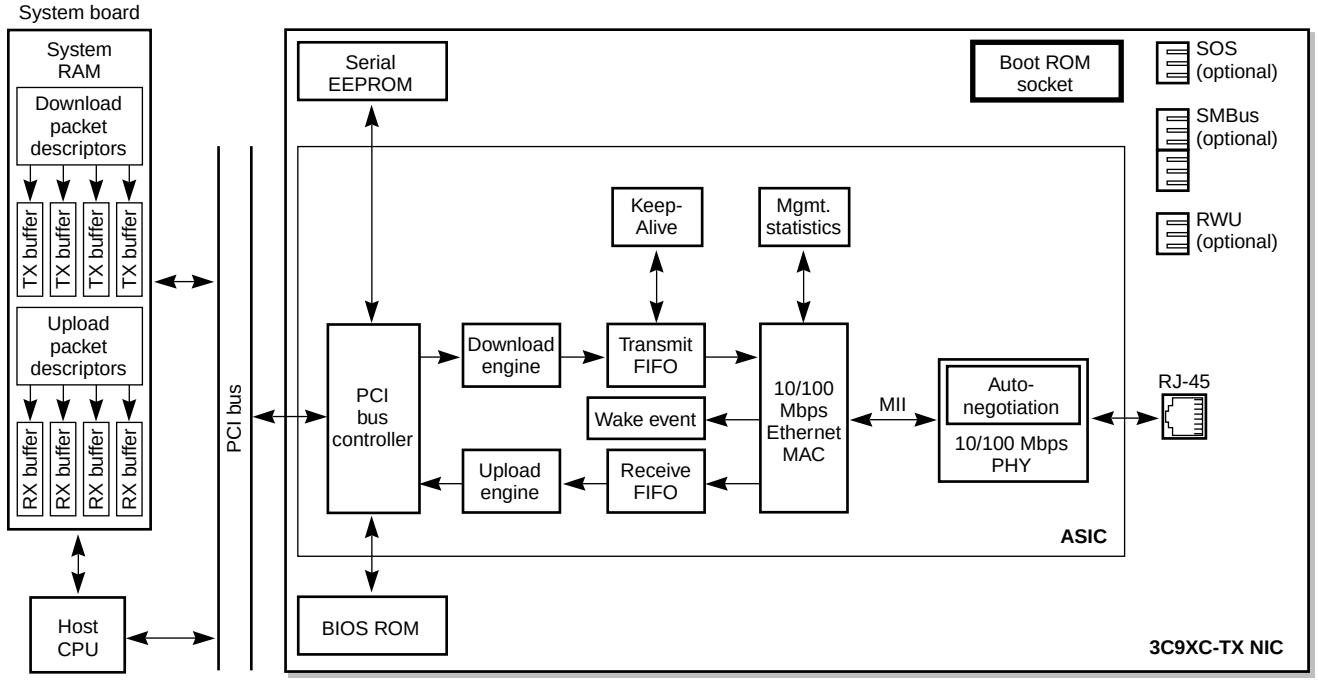
2 ARCHITECTURE

This chapter describes the 3C90xC NIC system architectures and ASIC block diagrams. It also summarizes the layout of the host registers and windows.

3C90xC NIC Block Diagram

The block diagram for the 3C90xC NICs is shown in Figure 1. The NIC devices are described at the end of this chapter.

Figure 1 3C90xC System Architecture



ASICs

Table 2 describes the 3C90xC ASIC versions.

Table 2 3C90xC NIC ASICs

ASIC Number	Description
40-0579-xxx	3.3-volt ASIC with an internal voltage regulator that allows it to operate in a 5-volt environment.
40-0574-xxx	3.3-volt ASIC with an internal voltage regulator that allows it to operate in a 5-volt environment.
40-05772-xxx	5-volt

Specifications in this technical reference apply to all ASIC versions unless the text designates a specific ASIC number.

Hardware Identification

To physically identify which ASIC is on a 3C90xC NIC, look at the ASIC number inscribed on the ASIC. (See Table 2 for a list of ASIC numbers.)

Software Identification

To identify through software which ASIC is on a 3C90xC NIC, view the chip/Vendor bit in the RevisionId register. See "RevisionId" in Chapter 4.

ASIC Block Descriptions

The devices associated with the ASICs are described below. For more information, see Figure 1.

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.

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. These blocks each contain 2 KB of data buffering and the logic required to manage the FIFOs.

The transmit FIFO, in addition to performing normal transmit operations, is also used to store both Wake-On-LAN patterns and keep-alive packets while in sleep mode. See Chapter 3 for specific operation.

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

This block provides the IEEE 802.3u auto-negotiation function.

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.

10/100 Mbps PHY

This block replaces the 10BASE-T/AUI, 100 Mbps signaling, and auto-negotiation functions found in 3C90x and 3C90xB NICs. It integrates the following functions:

- IEEE 802.3 Media Access Control (MAC) function. The protocol, includes deference, collision recovery and back off, receive packet filtering, and error detection. This block provides information to the management statistics function.
- 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 supports 10BASE-T, AUI, and thin coax (10BASE2) media types. Only filtering magnetics are required off-chip to implement a complete 10BASE-T solution.

Keep-Alive

Keep-alive packets allow a NIC that is in sleep mode (i.e., the PCI host is powered down) to transmit packets to refresh its presence in various network routing tables.

Wake Event

The NIC will generate a wakeup signal as a result of any of three standard wakeup events: Wakeup Packet reception, Magic Package reception, or change in link state.

SOS Connector

The SOS connector allows a system configuration where an external event (e.g., fan speed, over-temperature, over-voltage) causes a transmission of an alert packet over the network.

SMBus Connector

Attaching an SMBus controller to this connector allows:

- Arbitration with multiple SMBus master controllers
- Access to any ASIC register, including I/O and Configuration registers
- Issuing of commands (via the Command register)
- EEPROM read and write
- Reception Send and Receive packets
- Issuing of a System Wakeup (with assertRemotePme)
- Monitoring of network activity (through txActivity and rxActivity)

RWU Connector

The Remote Wake Up connector can be used to support WOL applications. Upon receiving a wakeup packet, a PME signal is generated, waking up the host system.

Other NIC Devices

The other devices associated with NIC operation are described below.

BIOS ROM

The optional BIOS ROM can contain up to 128 KB of code that is executed at system boot time.

NICs generally come with a BIOS ROM socket that allows field installation and upgrade of the boot code.

The following boot roms are supported:

- AT29C512
- AT49F010
- AT49BV512
- AT49F001N
- AT49LV001NT
- AT29LV512
- AT29LV010B
- SST29EE010
- SST29LE010
- SST39SF010
- SST39VF010
- SST39VF512
- STM29W512B
- AM29LV010B

Serial EEPROM

The 16-bit × 256-word and 16-bit × 1024-word serial EEPROM devices store configuration information for the NIC, including PCI device ID, station address, and transceiver selection. The type of EEPROM device installed is determined by a "power-on reset" register stuffing option. See the ResetOptions register for more information.

Host Registers

This section shows the host register layouts for the 3C90xC NIC.

The NIC interacts with the host CPU through registers. 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. A driver issues the SelectRegisterWindow command to the NIC to select which bank is visible in the window. The remaining 112 bytes in the register space are a flat address decode.

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, 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. Additionally, because of internal architecture limitations, the StationAddress register must be accessed with no larger than word-wide cycles.

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 and to determine which window is currently visible. IntStatus also includes a bit that indicates when a command issued to the Command register is in the process of being executed.

The 3C90xC 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.

3C90xC NIC Register Layout

The host register layouts and register window layouts for the 3C90xC NIC are shown in Table 3 and Table 4.

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

Table 3 3C90xC Host Register Layout

Byte 3	Byte 2	Byte 1	Byte 0	Offset
		PowerMgmtCtrl		7c
UpMaxBurst		DnMaxBurst		78
		DebugControl		74
DebugData				70
				6c
				68
				64
				60
				5c
				58
				54
				50
				4c
			ConfigData	48
			ConfigAddress	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 (See Table 4)				c
				8
				4
				0

Table 4 3C90xC 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		TriggerBits		c	1
	SosBits	WakeOnTimer		8	
SmbRxBytes		SmbDiag	SmbArb	4	
SmbStatus		SmbAddress	SmbFifoData	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 3C90xC NIC supports the PCI memory commands summarized in Table 5. The NIC decides on a burst-by-burst basis which command to use.

Table 5 3C90xC NIC 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.

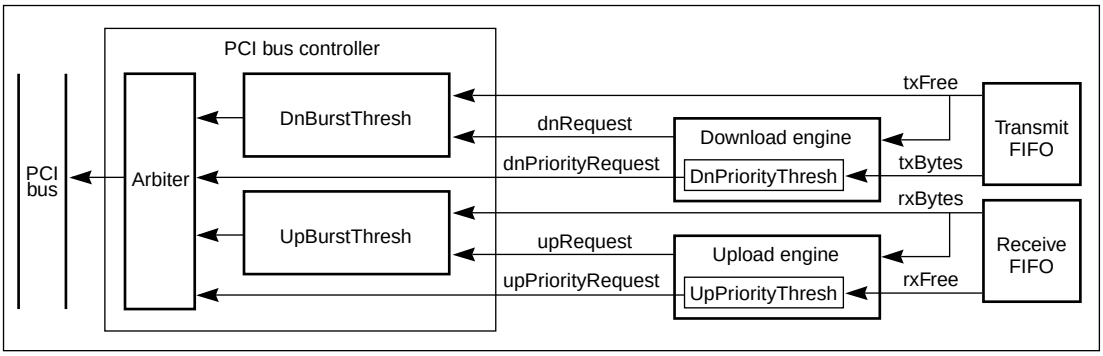
Three configuration bits control the use of advanced PCI memory commands:

- 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 allows the driver to disable the NIC's use of MWI. By default, MWI is enabled.
- The defeatMRL bit in the DmaCtrl register allows the driver disable the NIC's use of MRL. By default, MRL is enabled.

PCI Bus Request Control

The NIC provides 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 2 illustrates the bus request structure.

Figure 2 3C90xC NIC 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/dnRequest (upload and download alternate if both requests are present)

Power Management

The NIC supports 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 Up Sequencing

The *PCI 2.2 Bus Specification* dictates power restrictions for NICs that use the auxiliary 3.3-volt power supply. Specifically, the specification states that a NIC shall not use more than 20 mA while in the D3cold state if the NIC is not enabled to source the PME# system wake-up signal. (For more information on the D3 cold state, see Table 6.

Through a special power-up sequencing, the NIC determines whether it can operate in a normal mode or whether it must power up in a 20 mA mode (only auxiliary power present) by reading a control word from the EEPROM. If the NIC powers up in a 20 mA mode, it does not respond to incoming packets or SMBus activity. To recover from this mode, the PCI main power must be restored.

Low-power Mode

The 3C90xC NIC supports D3cold, a very low-power mode. If the NIC is receiving auxiliary current and it is not enabled as a wake-up device, the NIC goes into a 20 mA mode. In this mode, the NIC is essentially "dead" and does not respond to any cycles (through the PCI bus, network, or SMBus). To restore normal operation, assert PCI main power back to the PCI bus.

Power States

Table 6 defines the supported power states. The current power state is determined by the powerState field in the PowerMgmtCtrl register.

Table 6 3C90xC NIC 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 or SMBus cycles.
$D0_{active}$	0	This is the normal operational power state for the NIC. 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. The 3C90xC NIC supports both wake-up events and SMBus cycles in this 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 PciParm2 word in EEPROM. 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. The NIC's function in the D1 state is to recognize wake-up events and pass them on to the system by asserting the PME# signal on the PCI bus. The 3C90xC NIC also responds to SMBus cycles in this state.
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 PciParm2 word in EEPROM. 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. Similarly, the function of the NIC in the D2 state is to recognize wake-up events and pass them on to the system by asserting the PME# signal on the PCI bus. The 3C90xC NIC also responds to SMBus cycles in this state.
$D3_{hot}$	3	This is the full power-down state for the NIC. In $D3_{hot}$, the NIC loses all PCI configuration information except for the value in the powerState bit. 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. The NIC's function in the $D3_{hot}$ state is to recognize wake-up events and pass them on to the system by asserting the PME# signal on the PCI bus. The 3C90xC NIC also responds to SMBus cycles in this state.
$D3_{cold}$	N/A	This is the power-off state for the NIC from a bus point of view. The NIC has no function in this state unless auxiliary current is supplied. If auxiliary power is supplied, the NIC's function in the $D3_{cold}$ state is to recognize wake-up events and pass them on to the system by asserting the PME# signal on the PCI bus. The 3C90xC NIC also responds to SMBus cycles in this state. When power is restored, the system guarantees the assertion of hardware reset, which puts the NIC into the $D0_{uninitialized}$ state.

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*.

PowerMgmtCap

Synopsis	Provides information about the NIC's power management capabilities.
Type	Read-only
Size	16 bits
Offset from CapPtr	2

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.

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 the PCI Bus Power Management Specification adherence value: ■ 001 = Adheres to *PCI Bus Power Management Specification version 1.0.* ■ 010 = Adheres to *PCI Bus Power Management Specification version 1.1.*
[9]	d1Support	This read-only bit, when set, indicates that this device supports the D1 power state. The 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. The 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#). 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: ■ xxxx1: Power management events possible from D0. ■ xxx1x: Power management events possible from D1. ■ xx1xx: Power management events possible from D2. ■ x1xxx: Power management events possible from $D3_{hot}$.

(1 of 2)

PowerMgmtCap Bit Descriptions (continued)

Bit	Name	Description
		■ 1xxxx: Power management events possible from $D3_{cold}$.
The 3C90xC NIC supports wake-up events from all D-states. Bits [15:11] are loaded from the EEPROM PciParm2 word.		

(2 of 2)

PowerMgmtCtrl

Synopsis	Allows control over the power state and the power management interrupts.
Type	Read/write
Size	16 bits
Offset from CapPtr	4

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		

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: ■ 0 = State D0 ■ 1 = State D1 ■ 2 = State D2 ■ 3 = State D3 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.
[12:9]	dataSelect	These read/write bits select which power consumption value is to be reported when reading the PowerConsumption register. The NIC currently supports values of 0x0 to 0x7h, based on the following table:
	Value in dataSelect	**Data Reported**
	0,4	D0 Power Consumed/Dissipated
	1,5	D1 Power Consumed/Dissipated
	2,6	D2 Power Consumed/Dissipated
	3,7	D3 Power Consumed/Dissipated
	8-15	not supported

(1 of 2)

PowerMgmtCtrl Bit Descriptions (continued)

Bit	Name	Description
[14:13]	dataScale	These read/write bits define the scaling factor associated with the PowerConsumption register. These bits always read 0x10b, meaning that the value in the PowerConsumption register should be multiplied by 0.01, giving a result in units of Watts from 0 to 2.56.
[15]	pmeStatus	This read/clear bit is set to indicate a wake-up event has occurred. This bit is set regardless of the value in pmeEn. Writing a one to this bit clears it. Writing a zero has no effect.

(2 of 2)

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 in the host system. It also contains status bits that indicate what specific events have occurred.

The PowerMgmtEvent register is cleared by a LVDRst.

The enable bits in this register determine what types of events can cause the NIC to generate a wake-up event (interrupt) on the PCI bus. All enable bits are qualified with the pmeEn bit in the PowerMgmtCtrl configuration register. If the pmeEn bit is clear, then wake-up generation is disabled and these bits are ignored.

The event bits indicate that an actual wake-up event has occurred. These bits are masked by their corresponding enable bits above. If the enable bit is set, then the event bit can never become set. Once set, the event bits are cleared by a read to the PowerMgmtEvent register.

PowerMgmtEvent Register Format

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

PowerMgmtEvent Bit Descriptions

Bit	Name	Description
[0]	wakeupPktEnable	This read/write bit, when set, causes the NIC to generate a wake-up event due to wake-up packet reception.
[1]	magicPktEnable	This read/write bit, when set, enables the NIC to generate a wake-up event due to a Magic Packet frame reception.
[2]	linkEventEnable	This read/write bit, when set, enables the NIC to generate a wake-up event due to a change in link status (cable is disconnected or reconnected).

PowerMgmtEvent Bit Descriptions (continued)

Bit	Name	Description
[3]	wakeOnTimerEnable	This read/write bit, when set, enables the NIC to generate a wake-up event due to the WakeOnTimer register expiring.
[4]	wakeupPktEvent	This bit indicates that a wake-up event (which meets the reception criteria set by software) has been received.
[5]	magicPktEvent	This bit indicates that a Magic Packet frame has been received.
[6]	linkEvent	This bit indicates that a link status event has occurred.
[7]	wakeOnTimerEvent	This bit indicates that a wake-on-timer status event has occurred.
[8]	kapEn	This read/write bit, when set, places the transmit FIFO in the keep-alive mode. The KeepAliveTimer starts running, and the NIC sends keep-alive packets as programmed.
[9]	startKap	This read/write bit, when set, allows the keep-alive function to start, if the kapEn bit is set.
[10]	linkWpToKaEn	Link wake-up packet to keep-alive packet. Whenever a receive packet matches a wake-up pattern with a non-zerotrigSel value, a keep-alive packet that has the same trigSel value is transmitted, and the PMEN pin is not asserted.

Remote Wake-Up

The 3C90xC NIC supports Remote Wake-Up, the ability to remotely power on a PC that is in standby or suspend mode.

The NIC can generate a wake-up signal to the system as a result of any of the three standard wake-up events defined by the *Device Class Power Management Reference Specification — Network Device Class*:

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

The NIC can also cause a system wake-up based on two additional events:

- Wake-on-Timer
- Wake-On-SMB

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.

Wake-up Packets

The NIC 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. This can occur in any power state (D0 to D3); however, normal transmit and receive functions do not operate properly when this function is enabled. 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":

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 NIC can signal wake-up when it receives a Magic Packet frame from another station.

The Magic Packet technology, developed by Advanced Micro Devices, allows 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 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 can take place in any power state (D0 to D3).

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 NIC 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.

The change of link state can take place in any power state (Do to D3).

Wake-on-Timer

The NIC can signal a wake-up event based on the WakeOnTimer register. This register is programmed with a value from 1 to FFFFh. Setting the wakeOnTimerEnable bit in the PowerMgmtEvent register causes a PME# and a subsequent system wake-up. This event is cleared by writing the pmeStatus bit in the PowerMgmtCtrl register.

Wake-on-Timer events can take place in any power state (D0 to D3).

WakeOnTimer Register

Synopsis	Used to generate a wake-up event
Type	Read/write
Size	16 bits
Window	1
Offset	8

The WakeOnTimer register provides a means for the NIC to generate a system wake-up event. When this register is loaded with a non-zero value and the wakeOnTimerEnable bit is set in the PowerMgmtEvent register, the WakeOnTimer register is decremented once every five minutes. When the timer transitions from 0001 to 0000h, a PME wake-event is triggered.

This timer does not rearm itself; it must be reprogrammed after each use.

Wake-on-SMB

The NIC can signal a wake-up event through a direct register write through the two-wire SMBus interface. Setting the assertRemotePme bit in the SmbStatus register has the direct effect of issuing a bus PME# signal. This event is cleared by writing the pmeStatus bit in the PowerMgmtCtrl register.

Wake-on-SMB events can take place in any power state (D0 to D3).

The SmbStatus register is also visible through PCI bus accesses. For testing purposes, a PME# signal can be generated through a normal register write.

Programming Remote Wake-Up Events

This section describes the sequences involved in programming the 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. Issue a TxFifoBisect command and download keep-alive packets (if keep-alive packets are to be enabled).
7. Program the PowerMgmtEvent register to enable desired wake-up events.
8. 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.
9. 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.
10. 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 restoring power to the PCI bus (if returning from a $D3_{cold}$ state), and deasserting the PCIRST# signal.

   The rising edge of this reset causes the NIC to return to a D0 state (PCI configuration information is lost).
3. The system scans 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 and clears the pmeEn bit, causing PME# to be deasserted.
4. 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.

5 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).

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

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

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

9 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.

Keep-alive Packets

Keep-alive packets are a feature that allows a NIC that is in sleep mode (the PCI host is powered down) to transmit packets to refresh its presence in various network routing tables.

Keep-alive packets are transmitted periodically because of internal timer/counter logic. They may be invoked by received packets that match appropriately programmed patterns in a hash filter located with wake-up pattern matching logic. Wake-up pattern matches set bits in the TriggerBits register.

A keep-alive packet becomes an SOS packet if the appropriate sosEncoded bit field is set.

Installation

To prepare the transmit FIFO to accept keep-alive packets:

1 Disable the transmitter.

2 Send a TxFifoBisect command.

3 Download packets into the NIC.

Keep-alive packets are stored in the second half of the transmit FIFO RAM. They resemble normal transmit packets, except in the second dword of the Frame Start Header (FSH), which is downloaded into the NIC as the first dword of the first fragment of each keep-alive packet (the keep-alive packet control word).

The Frame Start Header for keep-alive packets is as follows:

Frame Start Header

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										

Keep-alive Control

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 last packet among the keep-alive packets must have a 1 set in the lastKap bit of the Frame Start Header.

Frame Start Header

Bit	Name	Description
[28]	rndupDefeat	This bit defeats packet length roundup.
[24]	lastKap	This bit distinguishes the last keep-alive packet in the sequence.
[23]	reArmEnable	When this bit is set, the transmit path does not disable itself upon seeing a max collision, but instead silently discards the packet. This bit is intended for keep-alive and SMBus packet transmits only.
[9:2]	pktId	This bit has no function in a keep-alive packet.
[1:0]	rndupBndry	This bit is the roundup boundary.

Keep-alive Control

Bit	Name	Description
[31:29]	sosEncoded	These bits encode one of seven external SOS pins. They indicate that this packet should be sent in response to an active low on the respective pin. A 000_2 in this field disables this feature. SOS pins two through seven are valid only as long as the NIC is not in the MII or MII External MAC modes, as defined in the InternalConfig register.
[28]	sendIfPciHot	This bit indicates that this packet should be sent only if the timer matches AND the PCI bus has power. If this bit is 0, the packet is sent.
[27:24]	trigSel	When this encoded field contains a non-zero value and it matches the respective bit in the TriggerBits register, the keep-alive packet is sent. If this field contains all zeros, it indicates that the keep-alive packet is not associated with the TriggerBits register.

SOS Name	sosEncoded	Tornado Pin
n/a	000	n/a
SOS[1]N	001	SOS
SOS[2]N	011	TXCLK
SOS[3]N	010	TXEN
SOS[4]N	100	CRS
SOS[5]N	101	RXOE
SOS[6]N	110	RXCLK
SOS[7]N	111	MDCLK

(1 of 2)

Keep-alive Control (continued)

Bit	Name	Description
[15:0]	KatTime	Each bit in this field represents a power of two seconds, which is compared to a non-visible KeepAliveTimer. When the timer and katTime match, this packet is sent. **Note:** Only one bit within this field should be sent The KeepAliveTimer value is reset by any of the following events: ■ Rising edge of PCIRst# ■ Falling edge of SOS[0]N or SOS[1]N ■ TxReset command with fifoTxReset bit not masked ■ kaEn is not set The following are some intervals for the KeepAliveTimer:

KatTime Value	period between transmissions
0000 0000 0000 0000	undefined, reserved
0000 0000 0000 0001	1 second
0000 0000 0000 0010	2 seconds
0000 0000 0000 0100	4 seconds
0000 0000 0000 1000	8 seconds
0000 0000 0001 0000	16 seconds
0000 0000 0010 0000	32 seconds
0000 0000 0100 0000	1.06 minutes
0000 0000 1000 0000	2.13 minutes
0000 0001 0000 0000	4.26 minutes
0000 0010 0000 0000	8.53 minutes
0000 0100 0000 0000	17.06 minutes
0000 1000 0000 0000	34.13 minutes
0001 0000 0000 0000	1.14 hours
0010 0000 0000 0000	2.28 hours
0100 0000 0000 0000	4.56 hours
1000 0000 0000 0000	9.12 hours

Note: If more than one bit is set, the behavior is undefined. If KatTime is zero, the packet is only to be sent due to wake-up pattern matching.

(2 of 2)

It is possible for the same keep-alive packet to be sent both in response to wake-up patterns and to matching a timer value in kaTime. However, in the unusual case that the wake-up event triggers a keep-alive sequence in which kaTime also matches, the packet is only sent once.

Activation To activate keep-alive operation:

1. Set kapEn in the PowerMgmtEvent register.
2. Issue the TxFifoBisect command.
3. Download the keep-alive packets.

   Make sure that the last one has the lastKap bit set.

4 Set startKap in the PowerMgmtEvent register to start the keep-alive timer.

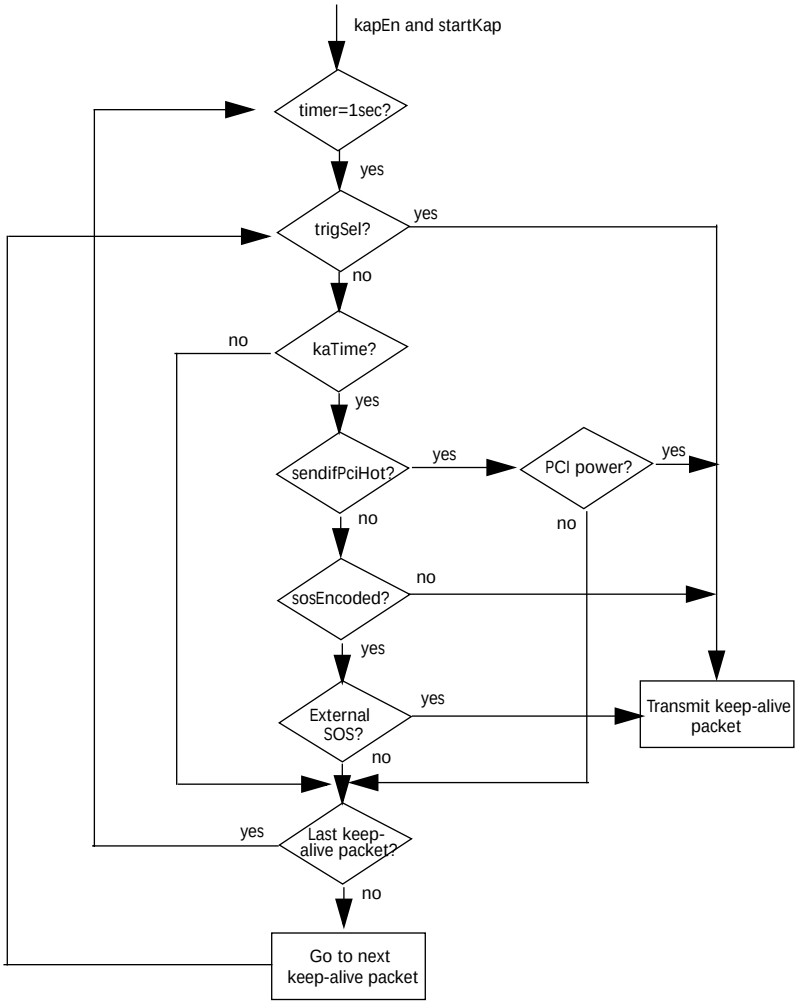
5 Change the power state to non-zero.

Transmission Timing

A nonvisable, almost free-running timer is used as the basis for all keep-alive transmissions. This timer goes off approximately every second, and all keep-alive packets are checked to see if they should be transmitted.

Figure 3 demonstrates when keep-alive packets are transmitted.

Figure 3 Keep-alive Packet Transmission Timing



Linked Wake-up Pattern to Keep-alive

If an incoming packet matches a loaded wake-up pattern and it has a respective keep-alive packet, that packet is sent on the next KeepAliveTimer pulse. If the pmeEn bit in the PowerMgmtCtrl register is set, a PME# wake-up signal is also asserted.

SOS

The 3C90xC NIC has an optional SOS connector with seven SOS pins that allow you to configure when an external event (such as fan speed, an over-temperature, or an over-voltage) asserts a level on one of these pins, which results in the transmitting of an alert packet over the network.

Table 7 describes the SOS pins.

Table 7 SOS Pins

SOS Pin	Description
SOS[1]N	External, active-low pin that is dedicated to an SOS alarm function.
SOS[2]N	TXCLK; active-low pin that can be used for SOS if its primary functions (MII internal or external) are not required in the application.
SOS[3]N	TXEN; active-low pin that can be used for SOS if its primary functions (MII internal or external) are not required in the application.
SOS[4]N	CRS; active-low pin that can be used for SOS if its primary functions (MII internal or external) are not required in the application.
SOS[5]N	RXOE; active-low pin that can be used for SOS if its primary functions (MII internal or external) are not required in the application.
SOS[6]N	RXCLK; active-low pin that can be used for SOS if its primary functions (MII internal or external) are not required in the application.
SOS[7]N	MDCLK; active-low pin that can be used for SOS if its primary functions (MII internal or external) are not required in the application.

A keep-alive packet transmission may be conditioned by any one of the SOS pins, or by none. The SOS pins contain weak, internal pull-up resistors, and therefore do not need to be connected to anything if they are not being used.

TriggerBits Register

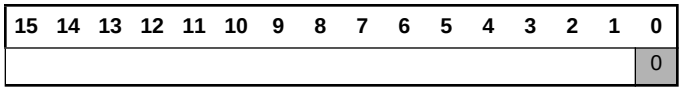
Synopsis	Indicates whether a wake-up packet has been received and whether its corresponding keep-alive packet has not yet been received.
Type	Read/write
Size	16 bits
Window	1
Offset	c

Each bit in the TriggerBits register indicates whether a wake-up packet has been received, and whether or not its corresponding keep-alive packet has been received. The bit position tells which HashTable entry matched the wake-up packet. There is no “zero” HashTable entry.

The value that is written into the TriggerBits register is ORed into the register. Zero values are ignored. Thus, by writing to this register you can simulate the arrival of a wake-up packet, as far as the keep-alive mechanism is concerned.

Each bit is cleared when its corresponding keep-alive packet is transmitted. The wake-up patterns are numerically ordered from 1 to 15, based on the order in which they were placed in the transmit FIFO.

TriggerBits Register Format



IEEE 802.3x Flow Control

The 3C90xC NIC supports 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 13 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.

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 hardware only flags an oversized packet on an IEEE 802.1Q frame if it is greater that MaxPktSize, plus four. (That is, the software does not need to change the default value of 1514.)

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.

TCP/IP Checksum Support

The 3C90xC NIC provides 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.

System Management Bus (SMBus) Interface

The 3C90xC NIC has a three-wire System Management Bus (SMBus) interface. The three signals are:

- SMBData
- SMBClock
- SMBChipSelectN

The SMBChipSelectN signal is an active-low signal that is used to qualify the entire SMB transaction.

The SMBus interface has a predefined transaction format through which an SMBus controller can:

- Arbitrate with multiple SMBus master controllers.
- Access any register within the ASIC. (This includes both I/O and Configuration registers.)

- Issue commands through the Command register.
- Read and write to the EEPROM.
- Send and receive packets.
- Issue a system wake-up with the assertRemotePme bit.
- Monitor relative network activity using the txActivity and rxActivity bits.

Transaction Format

The SMBus design supports two transaction types:

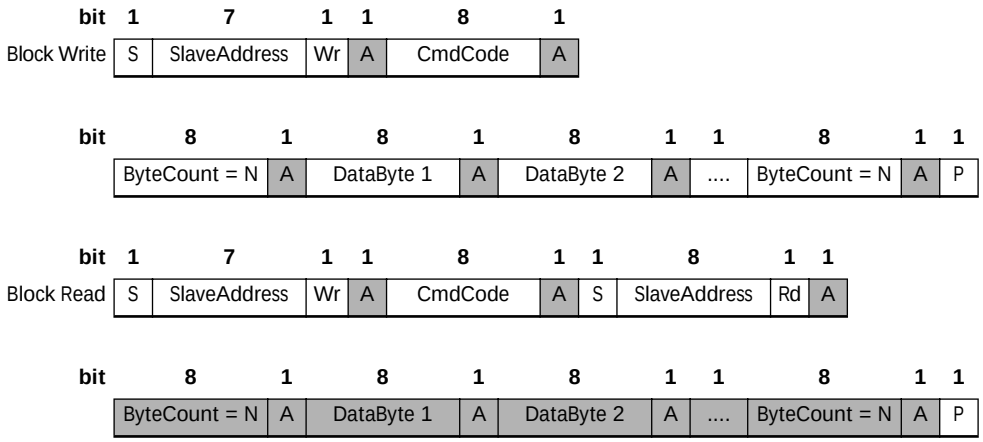
- Block Reads — The hardware infers the access size through the defined format and ignores the byte count sent by the SMBus master. Always access registers in the defined size.
- Block Writes — The hardware provides the byte count to the SMBus master based on the size of the accessed register, or in the case of a RxFifo read, 32 (as long as the bytes remaining are less than or equal to 32). When there are less than 32 bytes remaining, the byte count properly reflects this value.

Through the two transaction types, any access can be made to the I/O registers or to the transmit and receive FIFOs.

The ByteCount field is not used by the hardware to determine the size of the access for register-access cycles. However, it is used by TxFifo write and RxFifo read cycles.

The Block Read and Block Write transaction types are described below. Unshaded areas indicate Master to Slave; shaded areas indicate Slave to Master.

The SMBChipSelectN must be valid for the entire cycle.



Note:

- S — start
- Wr — write
- Rd — read
- A — ack
- P — stop

The SlaveAddress is compared to the seven bits of data this is stored in the SmbAddress register. This register is loaded from the EEPROM during power up, or it can be written through the PCI Host interface.

The CmdCode is the address that is offset into the 3C90xC NIC's windowed register map (see "3C90xC NIC Register Layout" in Chapter 2).

The ByteCount is the number of bytes in the Block Read or Block Write transaction type that follow. The hardware, however, calculates the transaction size based not on this number, but on the register size that is being pointed to by the CmdCode. Therefore, always access registers in their natural size, as defined in this technical reference.

Transaction Examples

```
setWindowSelect command to switch to Window#1

Wr = 1, CmdCode = OE, ByteCount = 2, DataByte1 = 01, DataByte 2 = 08

SmbStatus register access, (assumes Window#1 context)

Wr = 1, CmdCode = 02, Wr = 0, ByteCount = 2, DataByte1 = lowerByte,
DataByte2 = upperByte
```

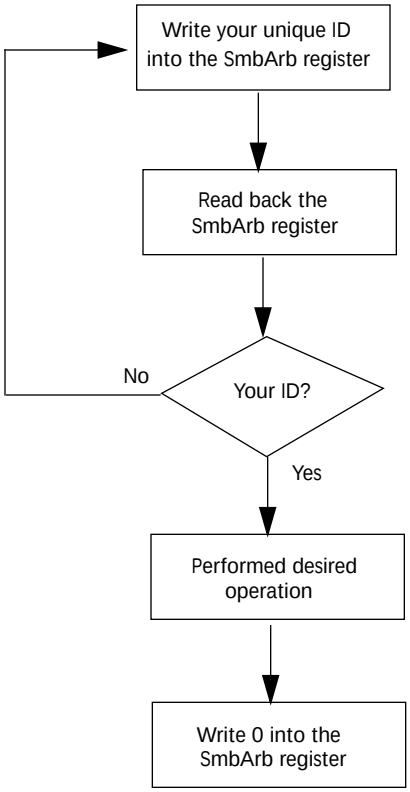
Multiple SMBus Master Arbitration

A mechanism is provided to arbitrate for locked access in systems where multiple SMBus masters coexist, and each needs access to the NIC's ASIC. (Note that this situation is rare. Typically, only one SMBUs master communicates with the NIC.)

This arbitration is performed through the SmbArb register. The SmbArb register is an eight-bit register that can only be written to with a non-zero value if it already contains a zero.

Obeying the SMBus Master Arbitration Flow protocol illustrated in Figure 4 ensures that only one SMBus master can access the NIC at any given time:

Figure 4 SMBus Master Arbitration Flow



Register Access Through the SMBus, any register within the NIC's ASIC can be accessed by the transaction format illustrated in Figure 4. This includes registers that exist within the PCI Config space (through the ConfigAddress and ConfigData registers located at offset 44h and 48h, respectively) and commands that are issued (which are essentially register writes).

Transmitting a Packet A packet can be sent over the SMBus by performing the following steps:

1. If normal PCI bus master operations are functioning, issue a DnStall command (3002).
2. Read the SmbStatus register and determine if operations have completed and conditions are right to transmit (okToXmit).
3. Form a four-byte Frame Start Header (FSH).

   See the Packet Download and Transmission sections in Chapter 6 for more information.
4. Append a properly formed Ethernet packet, including the source address, destination address, ethertype, and data information.

   The hardware "pads" out to 64 byes and appends the proper CRC to the end.
5. Switch the window context to window #1 (if necessary) by sending a SelectRegisterWindow command.
6. Issue multiple Block Write commands to download the above data into the transmit FIFO through the SmbFifoData register.

7 Issue a TxDone command (3800) to send the packet out onto the wire, flush any odd bytes, and align the transmit FIFO pointers for the next packet.

8 Restore normal operation (if applicable) by issuing a DnUnStall command (3003).

Receiving a Packet

The SMBus can receive normal packets, Magic Packet frames, or packets that have been pattern matched when the host is not in operation.

To receive normal packets:

1 Determine if a packet is in the receive FIFO by reading the InStatus register and looking for the rxComplete bit.

2 Issue multiple Block Read commands to upload the data from the receive FIFO through the SmbFifoData register.

The ByteCount returned by the Block Read command is 32 (as long as the bytes remaining to be uploaded are less than or equal to 32). If the SMBus controller wants to read more data, it should check the SmbRxBytes register to determine if there is more data in the packet. The SmbRxBytes register decrements to zero when all of the packet is read.

3 Issue an RxDiscard command after reading all or part of the packet (in blocks of 32).

To receive Magic Packet frames or pattern-matched packets:

1 Place the NIC in the smbPMEMask mode (bit [8] in the FifoDiagnostic register).

2 Determine if a Magic Packet frame or a pattern-matched packet is in the receive FIFO by reading the SmbStatus register and looking for the wakeEventPending bit.

3 Issue multiple Block Read commands to upload the data from the receive FIFO through the SmbFifoData register.

The ByteCount returned by the Block Read command is 32 (as long as the bytes remaining to be uploaded are less than or equal to 32). If the SMBus controller wants to read more data, it should check the SmbRxBytes register to determine if there is more data in the packet. The SmbRxBytes register decrements to zero when all of the packet is read.

4 Issue an RxDiscard command, or not, after reading all or part of the packet (blocks of 32).

Initiating a Keep-alive Packet

If the transmit FIFO already contains a set of keep-alive packets, the SMBus controller or the PCI host can cause one of the packets to be sent on the next KeepAliveTimer tick (approximately each second). This is done by setting a bit [15:1] in the TriggerBits register corresponding to the trigSel field in the keep-alive packets in the transmit FIFO.

A keep-alive packet is sent only once in response to a triggerBits bit being set. This bit is automatically cleared when the packet is sent.

Issuing a Wake-Up Event

The SMBus controller can cause its own wake-up event through the NIC as a result of a pattern-matched packet, a Magic Packet frame, or as an independent action. This gives the SMBUs controller the capability to authenticate, accept, or reject an external wake-up event.

The pmeEn bit needs to be set in the PowerMgmtCap register in the PCI Config space to pass the event to the PME signal.

Monitoring Network Activity

Two bits are provided in the SmbStatus register to infer network activity: txActivity and rxActivity. These bits are set by network transmit and receive traffic, respectively and are cleared by reading the SmbStatus register. The intent of these bits is to support network activity LEDs, or an equivalent function.

SmbAddress

Synopsis	SMBus address register.
Type	Read/write
Size	7 bits
Window	1
Offset	1

The SmbAddress register can be initialized through the EEPROM during power-up. It can also be ready or written to through the SMBus driver or PCI host software. A unique base address is needed for an SMBus slave to determine whether to respond to an SMBus master-initiated access.

SmbAddress Register Format

7	6	5	4	3	2	1	0
0							

SmbAddress Bit Descriptions

Bit	Name	Description
[0:6]	smbSlaveAddr	These bits contain the seven-bit slave address that is compared against the incoming SMBus master cycle to determine whether or not the cycle is for it.

SmbArb

Synopsis	Allows multiple SMBus masters to access the NIC's resources.
Type	Read/write
Size	8 bits
Window	1
Offset	4

SmbArb Register Format

7	6	5	4	3	2	1	0

SmbArb Bit Descriptions

Bit	Name	Description
[0:7]	smbMasterId	These bits are the SMBus master's unique ID.

The following steps are suggested for multiple SMBus masters to arbitrate the access of the NIC's resources:

1. Write its own ID with a non-zero value from an SMBus master.
2. Read arbitration register.

   If the same value is returned, this master wins the arbitration.
3. Follow the required procedures to access the NIC's transmit FIFO and receive FIFO for transmit and receive packets.
4. After completing the FIFO access, write the arbitration register with a zero ID.

The NIC's hardware ignores a non-zero ID write operation if the current ID is not zero.

SmbDiag

Synopsis	Used for hardware diagnostic purposes only.
Type	Read/write
Size	8 bits
Window	1
Offset	5

SmbDiag Register Format

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

SmbDiag Bit Descriptions

Bit	Name	Description
[0:3]	smbDiag	These read/write bits are defined by the SMBus IDs.

SmbFifoData

Synopsis	SMBus FIFO data register.
Type	Read/write
Size	8 bits
Window	1
Offset	0

SmbFifoData Register Format

7	6	5	4	3	2	1	0

The SmbFifoData register can only be accessed through the SMBus driver. A write access loads a byte of data from the SMBus to the transmit FIFO. A read access unloads a byte of data from the receive FIFO to the SMBus.

The SmbFifoData register can only be accessed through the SMBus controller. Accesses through the PCI bus are not supported.

The SmbFifoData register is not a "true" register in that no data actually resides in it. It is simply used as a decoded location when moving data in or out of the FIFOs.

SmbRxBytes

Synopsis	Provides the size information of a received packet.
Type	Read/write
Size	16 bits
Window	1
Offset	6

The SmbRxBytes register provides the size information of a received packet. The SMBus driver should read this register to determine the number of block transfers needed for uploading a received packet from the receive FIFO. The rxBytes bit is not decremented as the packet is being uploaded. It is only updated when the packet is discarded.

SmbFifoData Register Format

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

SmbArb Bit Descriptions

Bit	Name	Description
[0:12]	rxBytes	These bits are the number of bytes in the received packets.

SmbStatus

Synopsis	Indicates the status of the NIC through the SMBus to determine whether a packet can be transmitted or has been received.
Type	Read/write
Size	16 bits
Window	1
Offset	2

The SmbStatus register contains bits that indicate the status of various FIFO and network operations.

SmbStatus Register Format

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

SmbStatus Bit Descriptions

Bit	Name	Description
[0]	txError	This bit is an ORed version of soft error (max collision and transmit reclaim error) and hard error (jabber and transmit underrun).
[1]	linkDetect	This bit is a real-time indication of the link status.
[2]	autoNegComplete	This bit indicates whether auto-negotiation has completed successfully.
[3]	txActivity	This bit is set whenever the txInProg bit in the MediaStatus register is active. This indicates transmit activity. This bit is cleared by reading the register.
[4]	rxActivity	This bit is set whenever a packet is received into the receive FIFO (the fiRxComplete bit is true). This indicates receive activity. This bit is cleared by reading the register.
[5]	txEmpty	This bit indicates that the transmit FIFO is empty. The SMBus driver can poll this bit until the transmit FIFO is empty before downloading an SMBus packet or by issuing a txAgain command to retransmit a packet.
[6]	dnInProg	This bit indicates a PCI master download operation is in progress. The SMBus driver should issue a dnStall command and poll this bit until the dnInProg bit is inactive, before downloading a packet through the SMBus.
[7]	upInProg	This bit indicates a PCI master upload operation is in progress. The SMBus driver should issue an upStall command and poll this bit until the upInProg bit is inactive, before uploading a packet through the SMBus.
[8]	wakeEventPending	When PME mask mode is enabled (through the pmeMask bit [8] in the FifoDiag register), and either a pattern match or Magic Packet frame is detected, the PCI bus PME# signal is not asserted to wake-up the system. Instead, the wakeEventPending bit is set. The SMBus driver can poll this bit to determine whether a packet should be uploaded through the SMBus. This bit can be cleared by either writing a zero to it or by reading the PowerMgmtEvent register.

SmbStatus Bit Descriptions (continued)

Bit	Name	Description
[9]	assertRemotePme	This bit, when set to 1, asserts a PCI PME# signal. This signal should stay asserted until the system software writes a 1 to the pmeStatus bit [15] in the PowerMgmtCtrl register, which acknowledges that a wake-up event has occurred. The assertRemotePme bit is also cleared accordingly. The pmeEn bit in the PowerMgmtCtrl register must be set for this to take effect.
[10]	okToXmit	This bit, when set, indicates that the SMBus controller interprets that all conditions have been met and it is ready to download and transmit a packet.

(2 of 2)

4 CONFIGURATION

This chapter discusses the 3C90xC NIC configuration mechanism and defines the registers associated with configuration. Configuration has two components: NIC configuration and PCI configuration.

Power On Reset

Power On Reset (POR) occurs when the NIC is first powered on from a non-power state. (In PCs that supply auxiliary power to the NIC, POR occurs only when the AC is initially applied.)

On the rising edge of POR, the ASIC samples some of its pins and latches them into the ResetOptions register, setting various modes. See the ResetOptions register for more information.

System Reset

System reset is the assertion of the hardware reset signal on the PCI bus. For the 3C90xC NIC, however, this reset has limited affect on the ASIC.

The internal reset structure of 3C90xC NIC relies upon the ASIC Internal V_{DD}, creating a LVDRst signal that brings the ASIC to a known initial state. It is active when the ASIC Internal V_{DD} is less than approximately 2.9 volts.

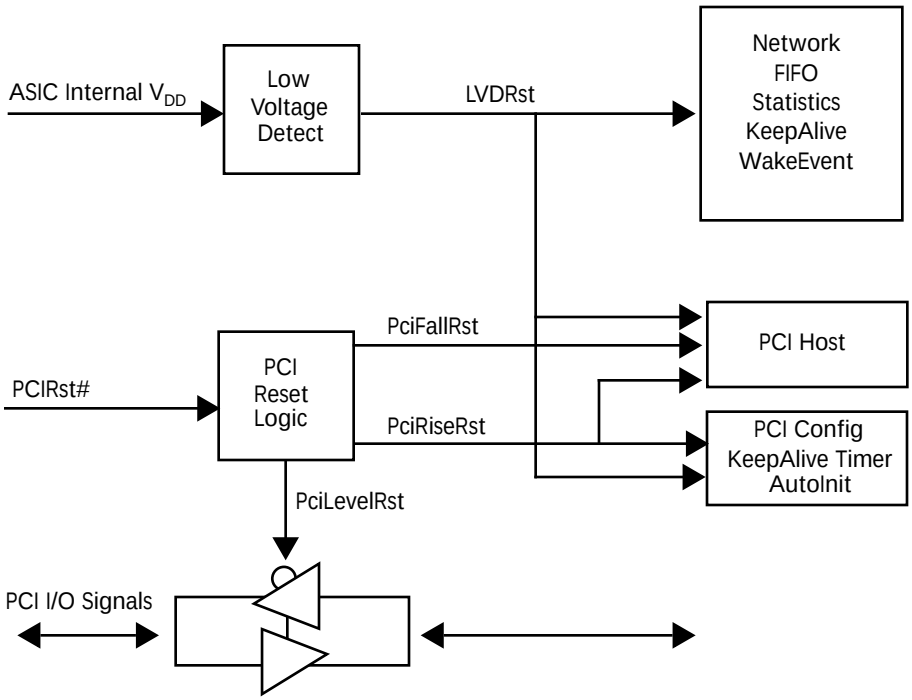
After the NIC is initialized, PCIRst#s are used only to reset sections of the PCI host logic; specifically, the PCI Target, PCI Master, PCI Config Space, and the KeepAlive Timer.

The PCIRst# consists of three logical signals:

- PciFallRst — generated on the falling edge of PCIRst#. This signal is a pulse of approximately 1 µs long and is intended to reset PCI bus activity as the bus is powered down.
- PciRiseRst — generated on the rising edge of PCIRst#. This signal is a pulse of approximately 1 µs long and is intended to reset PCI bus activity as the bus is powered up. It also resets the KeepAlive Timer, ensuring a known duration from reset until any subsequent keep-alive packet transmission.
- PciLevelRstb — a buffered version of the PCIRst# signal. This signal is used to disable the PCI I/O buffers during a bus reset condition.

Figure 5 illustrates the 3C90xC NIC internal reset structure.

Figure 5 Internal Reset Structure



Global Reset

A GlobalReset command is available for use by the driver software to reset the NIC. This command has a bit mask parameter that allows selective reset of various parts of the NIC.

See the Command register definition for more information.

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 8.

Table 8 EEPROM Data Locations

EEPROM Location	Register	Register Type
03	DeviceId	PCI configuration
08	PciParm	PCI configuration
0a	OEM Node Address Word 0	I/O

(1 of 2)

Table 8 EEPROM Data Locations (continued)

EEPROM Location	Register	Register Type
0b	OEM Node Address Word 1	I/O
0c	OEM Node Address Word 2	I/O
12	InternalConfig Word 0	I/O
13	InternalConfig Word 1	I/O
17	SubsystemVendorId	PCI configuration
18	SubsystemId	PCI configuration
19	MediaOptions	I/O
1b	SmbAddress	I/O
1c	PciParm2	PCI configuration
1d	PciParm3	PCI configuration

(2 of 2)

Flexible EEPROM Format

Data in locations 40h and above are interpreted as a flexible format. The format for this has a command-data structure, allowing any register to be written, command to be executed, or data to be loaded into the transmit FIFO.

In this manner, any arbitrary sequence of commands can be executed every time the EEPROM is loaded. It also allows packets to be loaded up into the transmit FIFO ready for transmission, which is useful when implementing SOS "failed to boot" messages out to the network. For more information, see Chapter 5.

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 3C90xC NIC responds only to Type 0 configuration cycles, directed at function 0. The NIC ignores Type 1 cycles and Type 0 cycles directed at functions other than 0.

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. 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 9.

Table 9 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 NIC includes a forced-configuration mode that enables it 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

Support for Signaling Standards

The 3C90xC NIC ASICs support the following signaling standards:

- The two physical signaling schemes defined in the IEEE 802.3u Fast Ethernet specification
- The 10 Mbps IEEE 802.3 10BASE-T signaling standard

Three basic media types are supported:

- 10 Mbps
- 100BASE-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.

On the 3C90xC NIC, the entire 100 Mbps PHY is integrated on-chip; there is no need for an external 100 Mbps transceiver. The MII registers are inside the 10/100 Mbps PHY and control not only the auto-negotiation function, but also various other functions within the PHY.

10 Mbps Signaling

10 Mbps signaling supports the 10BASE-T types of connection, this includes the Manchester encoder/decoder, clock recovery VCO, and the line drivers/receivers.

Magnetics connect the ASIC to the RJ-45 connector. With additional circuitry, 10BASE-T and 100BASE-TX can share the same RJ-45 connector.

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 3C90xC 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-TX solution.

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.

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.

On the 3C90xC NIC, 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 3C90xC NIC). 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.

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

BIOS ROM

The 3C90xC NIC supports 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:

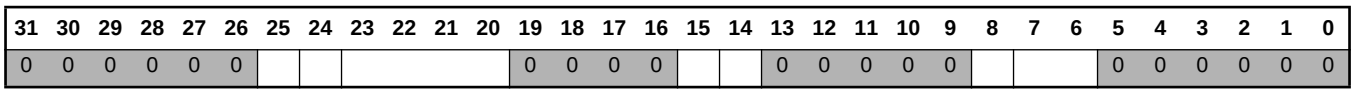
- InternalConfig word 0 corresponds to the low-order 16 bits of the register.
- InternalConfig word 1 corresponds to the high-order 16 bits of the register.

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.

InternalConfig Register Format



InternalConfig Bit Descriptions

Bit	Name	Description
[7:6]	romSize	Specifies the size of the BIOS ROM installed on the NIC, as defined below: ■ 0 = 64 KB ■ 1 = 128 KB ■ 2, 3 = Reserved
[8]	disableBadSsdDetect	This bit, when set, disables checking for a proper JK symbol in the first byte of the 100BASE-TX 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). This bit only has an effect in 100BASE-TX modes.
[14]	enableTxLarge	This read/write bit, when set, enables transmission of packets that are larger than the transmit FIFO. Because the FIFO size is 2 KB, this bit can be left clear (the reset default).
[15]	enableRxLarge	This read/write bit, when set, enables reception of packets that are larger than the receive FIFO. Because the FIFO size is 2 KB, this bit can be left clear (the reset default).

InternalConfig Bit Descriptions (continued)

Bit	Name	Description
[23:20]	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 MediaOptions register. After changing the value of xcvrSelect, drivers must issue both an RxReset and a TxReset command. 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. 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.
	1000	Auto-negotiation: The auto-negotiation function automatically determines the data rate and duplex mode. 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. 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. This setting enables a special operating mode in which the MII ASIC pins are used to expose the MAC/PHY interface signals.

InternalConfig Bit Descriptions (continued)

Bit	Name	Description
	1010–1111	Reserved.
[24]	autoSelect	When set, this bit indicates that the driver should ignore the value set in xcvrSelect, and instead automatically 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. 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 one of 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.

Selection Through 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 the 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.

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 NIC checks that bit 5 (autoNegComplete) in the MII register 01 (Status) 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.

   This involves doing a bitwise AND on bits [5:8] of MII registers 04 and 05 (Autonegotiation Advertisement 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.

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 StationAddress is loaded at power-up. The NIC's network address can be obtained from the appropriate data locations within the EEPROM. The driver can 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 the RxFilter register.

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 3C90xC NIC contains a hash filter for selective reception of multicast packets. The hash filter is an array of 256 enable bits.

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 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 "Data Format" 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.

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 (Autonegotiation Advertisement 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 10 summarizes the NIC PCI configuration registers. Shaded spaces and all locations within the 256-byte configuration space that are not shown in the table are either reserved or not implemented and return zero when read.

Table 10 Summary of PCI Configuration Registers

Byte 3	Byte 2	Byte 1	Byte 0	Offset
Data		PowerMgmtCtrl		e0
PowerMgmtCap		NextPtr	CapID	dc
				60 – d8
				5c
				58
				54
				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.

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

BiosRomControl 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	

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. 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.

CacheLineSize 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.

CapID The CapID register is at Offset 0 from the CapPtr register.

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

CapPtr This hard-coded value points to the beginning of a chain of registers that describe enhanced functions. This register returns 0xdc, which points to the power management registers.

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

Data The Data register is at Offset 7 from the CapPtr register.

The read-only Data register provides a mechanism to report the power consumed by the NIC in any of the D-states. The specific value and scaling is determined by the dataSelect and dataScale fields in the PowerMgmtCtrl register.

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.

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 10.

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 NIC always uses INTA#, so 01h is returned in InterruptPin.

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.

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.

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

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.

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.

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.

100 Mbps NICs return the value 5 in this field. 10 Mbps-only NICs return the value 48d.

MemBaseAddress 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.

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.

The 3C90xC NIC returns the value 5 in this field.

NextPtr The NextPtr register is at Offset 1 from the CapPtr register.

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.

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.
[7]	fastBackToBack	This read-only bit indicates that the NIC, as a target, supports fast, back-to-back transactions. 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.
[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. 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.

PciStatus Bit Descriptions (continued)

Bit	Name	Description
[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 provides a revision code for the ASIC.

RevisionId Register Format

7	6	5	4	3	2	1	0

RevisionId Bit Descriptions

Bit	Name	Description
[4:0]	revision	This field encodes the chip revision. Bits [1:0] are hardwired to 00b. Bits [4:2] are used to encode the revision (this allows pertinent revision data to also be visible in the NetworkDiagnostic register).
[7:5]	chip/Vendor	This field encodes the type of ASIC.

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.

Bits [6:2] are visible in asicRevision, to allow both revision and vendor information to be visible here.

SubsystemId This is the value read from EEPROM word 18h after system reset.

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

SubsystemVendorId This value is read from EEPROM location 17h after system reset.

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

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.

5 EEPROM

This chapter provides information about the EEPROM contents and registers.

Data Format

Table 11 summarizes the contents of the 3C90xC NIC's EEPROM.

Table 11 EEPROM Contents

Offset (hex)	Field Name	10/100 TX (hex)
00	3Com Node Address (word 0)	xxxx
01	3Com Node Address (word 1)	xxxx
02	3Com Node Address (word 2)	xxxx
03	DeviceId	9200
04	Manufacturing Data - Date	xxxx
05	Manufacturing Data - Division	00xx
06	Manufacturing Data - Product Code	xxxx
07	ManufacturerId	6d50
08	PciParm	2940
09	RomInfo	0000
0a	OEM Node Address (word 0)	xxxx
0b	OEM Node Address (word 1)	xxxx
0c	OEM Node Address (word 2)	xxxx
0d	Software Information	0010
0e	Compatibility Word	0000
0f	Software Information 2	00aa
10	CapabilitiesWord	72a2
11	Reserved	0000
12	InternalConfig Word 0	0000
13	InternalConfig Word 1	0180
14	Reserved	0000
15	Software Information 3	0000
16	LANWorks Data 1	0000
17	SubsystemVendorId	10b7
18	SubsystemId	1000
19	MediaOptions	000a
1a	LANWorks Data 2	0000
1b	SmbAddress	6300

Table 11 EEPROM Contents (continued)

Offset (hex)	Field Name	10/100 TX (hex)
1c	PciParm2	ffb7
1d	PciParm3	b7b7
1e-1f	reserved	0000
20	Checksum #1: 00-1f	00xx

(2 of 2)

Flexible Format

Table 12 summarizes the contents of the 3C90xC NIC's flexible EEPROM format.

Table 12 Flexible EEPROM Format

Offset (hex)	Field Name	10/100 TX (hex)
21-2f	reserved	0000
30	Current IP address bytes 1-0	xxxx
31	Current IP address bytes 3-2	xxxx
32-3f	SMBus (OEM-specific)	xxxx
40-3fe	Flexible format	xxxx
ff	Checksum#2*: 40-fe	00xx
3ff	Checksum #3†: 40-3fe	00xx

* Checksum #2 is only applicable if a 4K EEPROM is used.
† Checksum #3 is only applicable if a 16K EEPROM is used.

3Com Node Address

This field contains the 3Com node address for the NIC. It is *not* the field to be programmed into the StationAddress register. See "OEM Node Address" 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 (920 for 3C905C 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:

3C905C NICs

9200 EtherLink 10/100 PCI (TX)

3C980C NIC

9805 EtherLink Server 10/100 PCI (TX)

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. ■ 00 = 64K × 8 ■ 01 = 128K × 8 ■ 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]	pulsedPME	This bit, when set, indicates that the PME signal sourced by the ASIC is pulsed with a duration of approximately 100 µs when a wake event occurs.
[1]	lower1Meg	This bit provides the value for the mapLowerMeg bit in the MemBaseAddress register.

PciParm Bit Descriptions (continued)

Bit	Name	Description
[2]	disableMemBase	This bit, 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.
[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 1540h but is normally overwritten by the value in EEPROM.

The EEPROM values specify the following:

Bit	
pulsedPME	0
lower1Meg	0
disableMemBase	0
minGnt	0101b (2.5 µs)
maxLat	110000b (12 µs)

OEM Node Address

This field is loaded into the StationAddress register upon power-up. 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. ■ 00 = Reserved ■ 01 = Normal ■ 10 = Maximum network performance ■ 11 = Minimum CPU utilization
[7:6]	Reserved	Reserved, set to zero.
[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). Note the opposite polarities of the linkBeatDisable and linkBeatEnable bits.

Software Information Bit Descriptions (continued)

Bit	Name	Description
[15]	fullDuplex	Indicates to the host software whether it should configure the NIC for full-duplex operation. ■ 0 = Disable full-duplex operation. ■ 1 = Enable full-duplex operation.

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 3C90xC NICs. Table 13 summarizes the capabilities of NICs.

Table 13 3C90xC 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
14	supportsKeppAlives	0 or 1

The capabilities bits that are set for the 3C90xC NIC are described below.

Capabilities Word Bit Descriptions

Bit	Name	Description
[1]	supportsFullDuplex	Indicates that the NIC supports full-duplex media operation.
[5]	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.
[14]	supportsKeepAlives	Indicates that the NIC supports keep-alive packets.

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.

Software Information 2

This word contains additional information for drivers.

Software Information 2 Bit Descriptions

Bit	Name	Description
[1]	fixedBroadcastRxBug	This bit, when set, indicates that the NIC hardware includes a fix for the broadcast receive bug. This bug required a bit to be turned on in the multicast hash filter in order to receive broadcast frames.
[2]	fixedEndecLpbackBug	This bit, when set, indicates that the bug in the ENDEC loopback function has been fixed.
[3]	wolConnector	This bit, when set, indicates that the NIC has a 3-pin auxiliary Remote Wake-Up connector.

(1 of 2)

Software Information 2 Bit Descriptions (continued)

Bit	Name	Description
[4]	pmePulsed	BWOL only. This bit indicates that the PME# signal is asserted with a pulse rather than a level. This bit is meaningless if the wolConnector bit is 0. When wolConnector is 1, the value of this bit indicates the type of Remote Wake-Up PME# signal: ■ 0 = Level signal (as per the PCI specification) ■ 1 = Pulsed signal (the Dell implementation)
[5]	fixedMWIBug	This bit indicates that the Memory Write Invalidate (MWI) PCI command bug has been fixed and the MWI command should be enabled. ■ 0 = MWI command is not working properly ■ 1 = MWI command is working properly
[6]	wolAfterPowerLoss	This bit, when set to 0, indicates that the BIOS should place the NIC in a Magic Packet wake-up state after a power loss.
[7]	autoResetToD0	This bit, when set, indicates that the NIC automatically goes to a D0 state on a PCI bus reset.

(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. ■ 0000 = Generic MII ■ 0001 = 100BASE-T4 - MII ■ 0010 = 10BASE-T - MII ■ 0011 = 100BASE-TX - MII ■ 0100 = 10BASE-T - Auto-negotiation ■ 0101 = 100BASE-TX - Auto-negotiation ■ Others = Reserved for future combinations 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. This value is used by the driver only when both of the following conditions are true: ■ The autoSelect bit in the InternalConfig register is clear. ■ The xcvrSelect bit in the InternalConfig register is set to MII or auto-negotiation.
[15:4]	Reserved	Reserved for future information bits.

Lanworks Data 1

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

SubsystemVendorId 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. If this were an OEM NIC, the OEM's 2-byte code would be used.

SubsystemId This is the 2-byte subsystem ID. The upper byte of this field reflects a NIC configuration (for example, 10 would be "Tx" and the lower byte would designate a software revision).

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

Lanworks Data 2 This word is reserved for use by LANWorks to hold data related to their expansion ROMs.

SmbAddress This word is loaded into the SmbAddress register. It is used as the NIC's slave SMBus address.

PciParm2 This word is loaded into the NIC and controls various hardware functions related to PCI bus operation.

PciParm2 Bit Descriptions

Bit	Name	Description
[7:0]	d0/d1Power	This byte contains the power consumed by the NIC while the NIC is in a D0 or a D1 power state. The value is in watts/100.
[8]	janitorBit	This bit, when set, indicates to the hardware that this NIC is the designated wake-up device, and can consume up to 350ma from the auxiliary power supply. If this bit is cleared, the ASIC must consume no more than 20ma of current while in a $D3_{cold}$ sleep state.
[9]	d1Support	This bit indicates that the NIC supports the D1 power state. This bit is reflected in bit 9 of the PowerMgmtCap register.
[10]	d2Support	This bit indicates that the NIC supports the D2 power state. This bit is reflected in bit 10 of the PowerMgmtCap register.

PciParm2 Bit Descriptions (continued)

Bit	Name	Description
[15:11]	pmeSupport	This field indicates the power states from which the NIC is able to generate a power management event (assert PME#). 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 reflected in bits [15:11] of the PowerMgmtCap register and are defined as follows: xxxx1: Power management events possible from D0. xxx1x: Power management events possible from D1. xx1xx: Power management events possible from D2. x1xxx: Power management events possible from $D3_{hot}$. 1xxxx: Power management events possible from $D3_{cold}$.

PciParm3

This word is loaded into the NIC and sources the power consumed in the D3 and D2 states.

PciParm2 Bit Descriptions

Bit	Name	Description
[7:0]	d2Power	This byte contains the power consumed by the NIC while in a D2 power state. The value is in watts/100.
[15:8]	d3Power	This byte contains the power consumed by the NIC while in a D3 power state. The value is in watts/100.

PowerMgmtCtrl

This word is loaded into the PowerMgmtCtrl register in the PCI config space.

PowerConsumption

The lower byte of this word is loaded into the PowerConsumption register in the PCI configuration space. This number represents the number of watts consumed/100 during operation. No attempt is being made to differentiate between the different D-states.

Current IP Address

Locations 30h and 31h in the EEPROM are reserved for the current IP address that is assigned to the NIC in the following format:

Location	31		30	
IPAddress	255	255	255	255

SMBus - OEM Specific

These 16 words (from 30h through 3Fh) are allocated for OEM-specific use.

Flexible Format

The 3C90xC NICs have the ability to do the following:

- Load information into any register within the MAC portion of the ASIC (not including MII registers in the PHY/PMC or the EepromCommand and EePromData registers)
- Issue commands through the command/status register
- Modify the PCI config space
- Load data into the transmit FIFO

The mechanism to complete the tasks above is to have a generic "command-data" format to the EEPROM data starting at location 40h. The format is as follows:

Command

[15:13]	Encoded Command	111: Autoinit Done (that is, the EEPROM has completed loading). 110: Reserved for future expansion. 101: PCI Config Space Write. 100: Reserved for future expansion. 011: Register Write. 010: Reserved for future expansion. 001: TxFifo Write. 000: Reserved for future expansion.
[12]	Byte/Word	0: Byte Access 1: Word Access
[11]	Reserved	
[10:8]	eeCurWin	Temporarily forces the window that is being accessed.
[7:0]	eeAddress	Address within the EEPROM that is being accessed.
[9:0]	eeTxByteCount	Represents the number of bytes following (word-justified) a TxFifo Write command.

Data

[15:0]	eePromData	This data is "right-justified" if it contains only a byte. For TxFifo commands, this is also the case for the last transfer of an odd length.

Checksum #2

This checksum for the EEPROM contents is a standard 16-bit TCP/IP algorithm that is computed across all bytes in EEPROM words 000 through 0feh and is written into the word at location 0ffh.

This is used only if a 4K EEPROM is connected to the ASIC.

Checksum #3

This checksum for the EEPROM contents is a standard 16-bit TCP/IP algorithm that is computed across all bytes in EEPROM words 000 through 3feh and is written into the low-order byte of word 3ffh.

This is used only if a 16K EEPROM is connected to the ASIC.

EepromCommand

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

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. In addition, the EEPROM's WriteEnable, WriteDisable, EraseAll, and WriteAll commands can be issued. The EepromCommand register defaults to 0000h upon reset.

Support for various size EEPROMs is listed as follows:

- 2K-bit (128x16)
- 4K-bit (256x16)
- 16K-bit (1024x16)

The command formatting required for 2K, 4K, and 16K differ in which bits the opcode and subopcode reside; however, to preserve legacy software's functionality, the hardware does the appropriate bit swapping and is therefore transparent to the software. This explains why the eepromOpcode is located in bits [7:6].

EepromCommand Register Format

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

EepromCommand Bit Descriptions

Bit	Name	Description
[11:8], [5:0]	eepromAddress	These read/write bits logically translate to a linear 1024 locations, identifying which 16-bit word is the target for the ReadRegister, WriteRegister, and EraseRegister commands. When the eepromOpcode bits [7:6] are 00_2, bits [5:4] are defined to identify an individual command among the following group of four subcommands: ■ 00 = WriteDisable (60 us) ■ 01 = WriteAll (11 ms) ■ 10 = EraseAll (11 ms) ■ 11 - WriteEnable (60 us)

EepromCommand Bit Descriptions (continued)

Bit	Name	Description
[7:6]	eepromOpcode	These read/write bits specify one of three individual commands and a single group of four subcommands. ■ 00 = WriteEnable, WriteDisable, WriteAll, EraseAll subcommands ■ 01 = WriteRegister (11 ms) ■ 10 = ReadRegister (162 us) ■ 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.

Two-bit opcodes and up to a 10-bit address are written into this 16-bit register to cause the NIC to carry out the desired EEPROM command.

The 3C90xC NIC ensures backward-compatibility only if the legacy software writes the EepromCommand register as a word.

If data is written to the EEPROM, the 16-bit data word must be written to the EepromData register by the host prior to issuing the associated write command. Similarly, if data is to be read from the EEPROM, the read data is available through the EepromData register 162 us after the ReadRegister command has been issued.

A mechanism within the EEPROM interface automatically disables writes and erasures to prevent accidental 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 prior to 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 one. Therefore, and EraseRegister or EraseAll command must be issued prior to attempting to write data to the EEPROM.

The EEPROM is a particularly slow device. It is important that the host wait until the eepromBusy bit is false before issuing 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 = xxxx xxxx 0011 $xxxx_2$
3. Verify that the eepromBusy bit is false.
4. Issue an EraseRegister command; opcode xxxx aaaa 11aa $aaaa_2$
5. Verify that the eepromBusy bit is false.
6. Write data pattern to the EepromData register.
7. Issue a WriteEnable command; opcode xxxx xxxx 0011 $xxxx_2$
8. Verify that the eepromBusy bit is false.
9. Issue a WriteEnable command; opcode xxxx aaaa 01aa $aaaa_2$

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 register for use with the NIC's serial EEPROM. Data that is read out of the EEPROM can be read by the host from this register when the eepromBusy bit becomes false. Data to be written to the EEPROM is written to the EepromData register prior to issuing the write command to the EepromCommand register.

Portions of the EEPROM are used for dynamic information. That is, there are locations that are written many times over the life of the NIC. Although the number of write cycles to a give location is very large (on the order of 1,000,000 cycles), it is not infinite. Care should be taken to not needlessly write to the EEPROM. For example, read a location first to see if the information has actually changed prior to writing out a value.

The EepromData register 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.

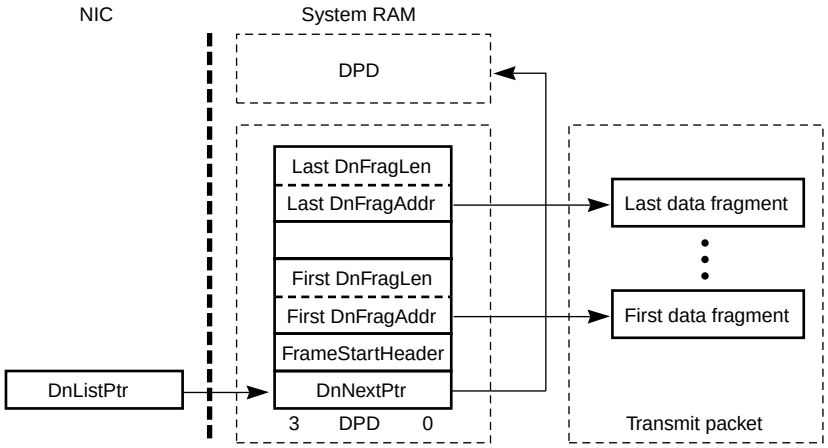
For information regarding keep-alive packets (special packets that can be transmitted while the NIC is in a sleep or suspend state) see "Keep-alive Packets" in Chapter 3.

The 3C90xC NIC supports a multipacket, multifragment gather process, whereby descriptors representing packets can be built in system memory and linked together by the host. The NIC follows 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 6.

Figure 6 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.

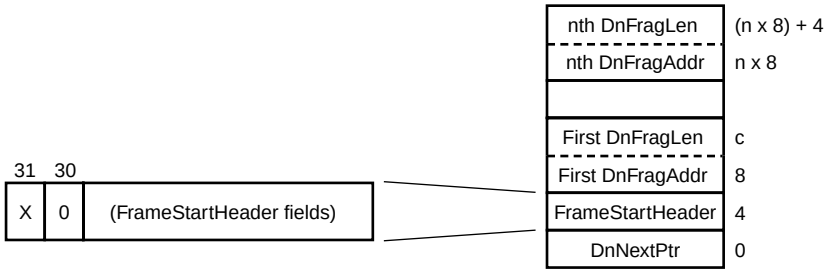
The NIC exits reset with the download engine in the idle state, ready to start processing a downlist as soon as a nonzero value is written into DnListPtr.

DPD Data Structure

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

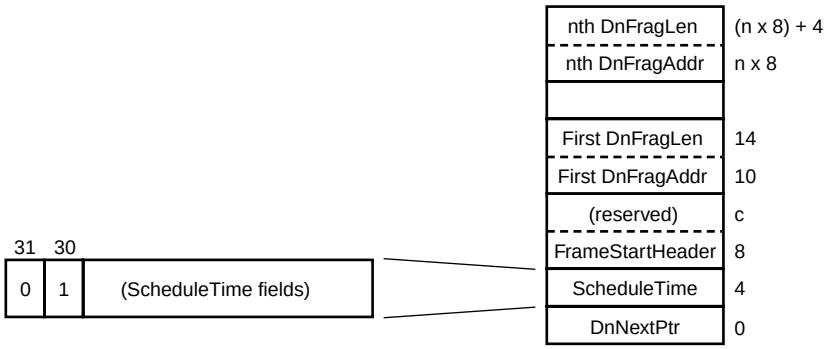
The Type 0 DPD format is shown in Figure 7. This format is backward-compatible with 3Com 3C90x NICs but has some added extensions.

Figure 7 Type 0 DPD Format



The Type 1 DPD format is shown in Figure 8. This format adds fields to support time scheduling of packet downloads.

Figure 8 Type 1 DPD Format



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

Bits [31:30] in the dword at offset 4 determine whether the DPD format is Type 0 or Type 1. Bit combination 11b is reserved for a future DPD format. If bits [31:30] are 01, then a ScheduleTime is inserted into the structure and all entries are shifted by 4.

Keep-alive packets have a slightly different Frame Start Header format. See "Keep-alive Packets" in Chapter 3 for more information.

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.

Frame Start Header

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

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										

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. ■ 00 = Up to dword ■ 10 = Up to word ■ x1 = No rounding up is performed
[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.
[13]	crcAppendDisable	The driver sets this bit to inhibit the NIC from appending a CRC to the end of this packet. 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. When this bit is cleared, the NIC computes and appends CRCs for transmit packets.
[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.

(1 of 2)

FrameStartHeader Bit Descriptions (continued)

Bit	Name	Description
[23]	reArmEnable	When this bit is set, the transmit path does not disable itself upon seeing a max collision; instead, it silently discards the packet. This is intended for keep-alive and SMBus packet transmits only.
[24]	lastKap	This bit distinguishes the last keep-alive packet in the sequence. This bit is only applicable when the keep-alive function is running.
[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. 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. 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. The NIC reads this bit after the download operation has finished, allowing the host to change this bit while the download is in progress. 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 contains a field defining when the packet should be downloaded.

The Alternate Download Sequence does not support the Schedule Time DPD structure.

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	1			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. 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. 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. 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. 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

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

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

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													

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.

DnFragLen Bit Descriptions (continued)

Bit	Name	Description
[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:
  - 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.

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.

   Because 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 3C90xC NIC has 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.

The NIC performs length roundup in a way that is compatible with older 3Com 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. 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.

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.

Adding DPDs to the End of the Downlist

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

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 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 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

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 3C90xC 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.

NIC Download Sequence

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

Original Download Sequence

Starting with the driver writing the DnListPtr register (for example, when starting from an empty downlist) the 3C90xC NIC follows this sequence:

1. Checks that the DnListPtr register is nonzero.
2. Checks that it is not in the DnStall command state.
3. Fetches 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. Fetches the DnFragAddr and DnFragLen DPD entries one by one from the DPD, and move the associated data fragments to the transmit FIFO.
5. If a transmit underrun occurs, waits until the driver issues a TxReset command.
6. Sets the dnComplete bit in the DPD.
7. If a DnStall command has been issued, waits until a DnUnStall command is issued.
8. Fetches the FrameStartHeader again, and, if the dnIndicate bit is set, sets the dnComplete indication (which may in turn cause an interrupt if the IndicationEnable and InterruptEnable masks are set correctly).
9. Fetches 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.

Alternate Download Sequence

Based on a dnAltSeqDisable bit in the DmaCtrl register, steps 6 through 10 in the Original Download Sequence are reordered to become:

6 If a DnStall has been issued, waits until a DnUnStall is issued.

7 Fetches the DnNextPtr from the current DPD.

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

9 Sets the dnComplete bit in the 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.
- 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

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

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

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. This bit is different from the upPktComplete bit in the UpPktStatus register in that upComplete latches on once an upPktComplete indication has occurred. 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. 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. 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. 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. If this bit is clear, Countdown expiration does not set the intRequested bit. If this bit is set, expiration of Countdown sets intRequested. 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. 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. 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. 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). 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. 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. For more information on the Countdown modes, see "Countdown" in Chapter 12.
[16]	upAltSeqDisable	Setting this bit disables the alternate upload sequence, so that the NIC mimics the behavior of earlier-generation (3C90x) NICs. 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.
[17]	dnAltSeqDisable	When this bit is cleared (default), the alternate download sequence is enabled (the sequence becomes: fetch next pointer, generate interrupt, write dnComplete). When this bit is set, the alternate download sequence is disabled (the sequence becomes: write dnComplete, fetch next pointer, generate interrupt).
[20]	defeatMWI	Setting this read/write bit prevents the bus master logic from using the Memory Write Invalidate (MWI) PCI command.
[21]	defeatMRL	Setting this read/write bit prevents the bus master logic from using the Memory Read Line (MRL) PCI command.
[22]	upOverDiscDisable	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. 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. 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 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, 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.

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

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 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 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

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.

TxPktId

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

TxPktId Register Format

7	6	5	4	3	2	1	0

The TxPktId 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.

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 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

7	6	5	4	3	2	1	0
							0

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 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.
[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.

7 RECEPTION AND UPLOAD

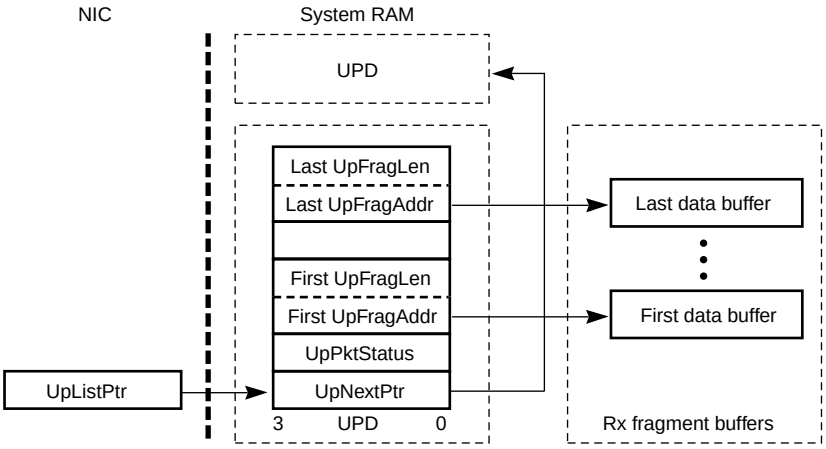
This chapter presents an overview of the packet reception process and defines the registers associated with the reception and uploading of data.

The 3C90xC NIC supports 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 9.

Figure 9 Uplist



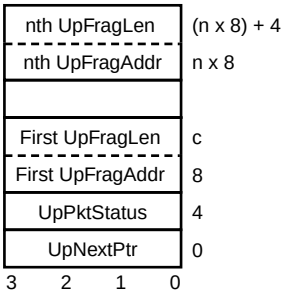
The head of the uplist is the UPD that corresponds to the current upload packet. The Up List Pointer (UpListPtr) register points to this UPD. As the UPD is processed, the fragment address and fragment length values are fetched one by one from the UPD into on-NIC registers, which are used to control the 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 10.

Figure 10 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. 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. 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 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 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 indicates that the packet contained an error in the UDP header checksum. This bit is only valid when the udpChecksumChecked bit is set.
[28]	impliedBufferEnable	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. 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). 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. 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, when set, indicates that the packet contained an IP header. The ipChecksumError bit in the UpPktStatus register contains the result of the checksum comparison.

UpPktStatus Bit Descriptions (continued)

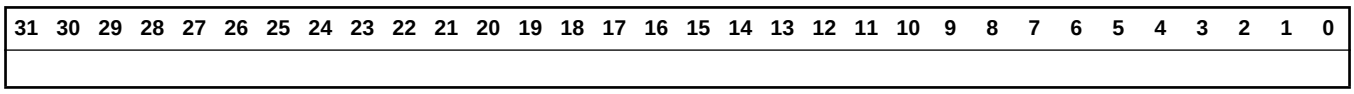
Bit	Name	Description
[30]	tcpChecksumChecked	This bit, when set, indicates that the packet contained a TCP header recognizable by the NIC (fragmented TCP datagrams do not set this bit). The tcpChecksumError bit in the UpPktStatus register contains the result of the checksum comparison.
[31]	udpChecksumChecked	This bit, 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

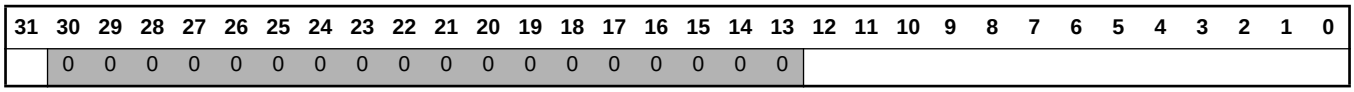


A fragment can start on any byte boundary.

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



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 3C90xC NIC comes out of reset with reception disabled.

Until reception is enabled, no incoming packets are accepted by the NIC. Once reception 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.

See "RxEarlyThresh" later in this chapter for more information.

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													

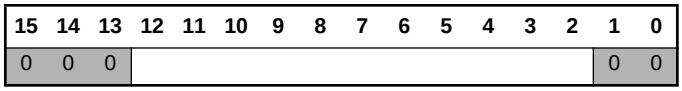
The value in MaxPktSize determines the minimum receive packet size that is flagged as oversize.

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



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.

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.

The rxEarly interrupt is meant to be usable as a retriggerable interrupt. It is legal for the driver to respond to an rxEarly interrupt due to a value set in the RxEarlyThresh register, 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 indication is generated as soon as the rxEarly threshold is crossed (or immediately if the threshold was already crossed).

RxFilter

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

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.

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.

(1 of 2)

RxFilter Bit Descriptions (continued)

Bit	Name	Description
[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	Setting this bit enables the NIC to receive packets that pass the hash filtering mechanism.

(2 of 2)

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.

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.

The 3C90xC NIC loads the StationAddress register from the EEPROM; however, because of a logic bug, the address is not byte-swapped as required. The drivers should continue to load these values as part of their initialization sequence and not rely on the hardware state machine.

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.

The StationMask is reset to all zeros on LVDRst.

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 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 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.

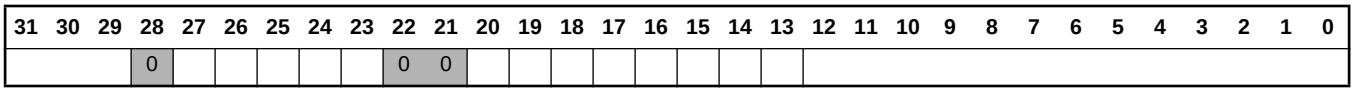
UpPktStatus

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

UpPktStatus 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.

UpPktStatus Register Format



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 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. 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. This bit is cleared with an UpUnStall command.
[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.

UpPktStatus Bit Descriptions (continued)

Bit	Name	Description
[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	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 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 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 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, when set, indicates that the packet contained an IP header, and ipChecksumError contains the result of the checksum comparison.
[30]	tcpChecksumChecked	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, 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.

(2 of 2)

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 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 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

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 the 3C90xC NIC, 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 14.

Table 14 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.
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 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.

The windowNumber in bits [15:13] reflects the current window pointed to. Separate values are maintained for SMBus accesses and PCI bus accesses.

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. This bit is acknowledged by issuing the AcknowledgeInterrupt command with the interruptLatchAck bit set.

IntStatus Bit Descriptions (continued)

Bit	Name	Description
[1]	hostError	This bit is set when a catastrophic error related to the bus interface occurs. The errors that set this bit are PCI target abort and PCI master abort. 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. 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. 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.
[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. 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). 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 expiration of the Countdown register. 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. A driver should respond to an updateStats interrupt by reading all of the statistics. This has the side effect of acknowledging (clearing) updateStats.

IntStatus Bit Descriptions (continued)

Bit	Name	Description
[8]	linkEvent	This bit indicates a change in the link status, as detected by the on-chip auto-negotiation logic. 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. 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. The linkEvent interrupt is cleared either by reading the IntStatusAuto register or by issuing an AcknowledgeInterrupt command with the appropriate bit set. 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.
[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. This bit is acknowledged by an AcknowledgeInterrupt command with the dnComplete bit set. 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.
[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. 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. These bits are reset after a hardware reset or a GlobalReset command.

(3 of 3)

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

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, interruptLatch, and linkEvent.

9 STATISTICS AND DIAGNOSTICS

This chapter provides an overview of statistics and defines the registers associated with statistics and diagnostics.

The 3C90xC NIC includes 12 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 15 and Table 16 and are described in alphabetical order in this chapter.

Table 15 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 16 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 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

The BadSSD 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 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.

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

The BytesRcvdOk 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

The BytesXmittedOk 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				

The CarrierLost 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

The FramesDeferred 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

The FramesRcvdOk 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 in the UpPktStatus register:

- upOverrun
- upRuntFrame
- upAlignmentError
- upCRCError
- 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 affect 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

The FramesXmittedOk 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

The LateCollisions register counts the number of late collisions. Because 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

7	6	5	4	3	2	1	0

The MultipleCollisions 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

The RxOverruns 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

The SingleCollisions 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				

The SqeErrors 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

The UpperBytesOk 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.

UpperBytesOk Register Format

7	6	5	4	3	2	1	0

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.

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

The UpperFramesOk 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.

UpperFramesOk Register Format

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

UpperFramesOk Bit Descriptions

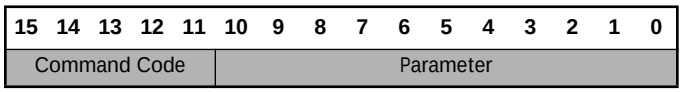
Bit	Name	Description
[1:0]	upperFramesRcvdOk	The high-order two bits of the FramesRcvdOk register. This value is latched whenever FramesRcvdOk is read.
[5:4]	upperFramesXmittedOk	The high-order two bits of the FramesXmittedOk register. This value is latched whenever FramesXmittedOk is read.

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

Command Register 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.

Summary of Commands

Table 17 summarizes the commands that are used by the 3C90xC NIC. The commands are described in alphabetical order in the following sections.

*Commands in Table 17 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 17 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.
	SetTxStartThresh	(1001 1••• •••• ••••)	Set the value of the TxStartThresh register.
	TxAgain	(1001 0XXX XXXX XXXX)	Retransmit the last packet in the queue that was just sent out of the transmit FIFO.
	TxDisable	(0101 0XXX XXXX XXXX)	Disable packet transmission.
	TxDone	(0011 1XXX XXXX XXXX)	Used by the external SMBus controller to signal to the NIC that the data which has been downloaded to the transmit FIFO is a complete packet.
	TxEnable	(0100 1XXX XXXX XXXX)	Enable packet transmission.
	TxFifoBisect	(1101 1XXX XXXX XXXX)	Logically split the transmit FIFO into two 1K FIFOs to prepare the transmit FIFO to accept keep-alive frames.
Receive	RxDisable	(0001 1XXX XXXX XXXX)	Disable packet reception.
	RxDiscard	(0100 0XXX XXXX XXXX)	Used by the external SMBus controller to cause the top receive packet to be discarded.
	RxEnable	(0010 0XXX XXXX XXXX)	Enable packet reception.
	SetHashFilterBit	(1100 1•XX XX•• ••••)	Program a particular bit in the hash filter.
	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.
	EnableDcConverter	(0001 0XXX XXXX XXXX)	Enable the 10BASE2 DC-DC converter.
	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.

Unused Command Codes

The following are command codes that are not used by the 3C90xC NIC:

- 10010
- 10100
- 11010
- 11100
- 11101
- 11110
- 11111

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.
[1]	endecReset	This bit, when set, masks reset to the internal 10 Mbps encoder/decoder.
[2]	networkReset	This bit, when set, masks reset to the network interface logic, including the CSMA/CD core, and the statistics registers. In 3C905B (ASIC 40-0476-00x or 40-0483-00x) and 3C90xC NICs, this bit masks reset 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. If this bit is not set, the EEPROM data is reloaded.
[5]	hostReset	This bit, when set, masks reset to the bus interface logic. If this bit is not set, the following registers are cleared: IntStatus, InterruptEnable, IndicationEnable, and Countdown.
[6]	smbReset	This bit, when set, masks reset to the SMBus interface.
[7]	vcoReset	This bit, when set, masks reset to the on-board 10 Mbps VCO.
[8]	upDownReset	This bit, when set, masks reset to the upload/download logic. 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.
[1]	endecRxReset	This bit, when set, masks reset to the internal Ethernet encoder/decoder receive logic.
[2]	networkRxReset	This bit, when set, masks reset to the network interface receive logic, including the CSMA/CD core. In 3C905B (ASIC 40-0476-00x or 40-0483-00x) and 3C90xC NICs, this bit masks reset to the PHY. If this bit is not set, the receiver is disabled, and RxFilter is cleared.
[3]	fifoRxReset	This bit, when set, masks reset to the receive FIFO control logic. 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. 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.

Bit Value		(0101 100• 0000 ••••)
Bit	**Name**	**Description**
[1]	endecTxReset	This bit, when set, masks reset to the internal Ethernet encoder/decoder transmit logic.
[2]	networkTxReset	This bit, when set, masks reset to the network interface transmit logic, including the CSMA/CD core. If this bit is not set, the transmitter is disabled, and the TxStatus queue is cleared.
[3]	fifoTxReset	This bit, when set, masks reset to the transmit FIFO control logic. 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. 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.

A DnStall may be issued by the SMBus controller through the SMBus interface. This stall condition is "OR"ed with the one from the PCI bus. Both conditions must be cleared by there respective DnUnstall to continue normal operation.

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.

This command must also be issued by the SMBus controller if it had previously issued a DnStall command to resume normal operation.

SetTxReclaimThresh

Bit Value	(1100 0000 •••• ••••)

The SetTxReclaimThresh command sets the TxReclaimThresh register to the desired value.

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.

TxAgain

Bit Value	(1001 0XXX XXXX XXXX)

The TxAgain command retransmits the last packet in the queue that was just sent out of the transmit FIFO. The transmit FIFO must be empty when issuing the TxAgain command.

The intended use of this command is to allow an SMBus controller to be able to send out multiple copies of the same packet, simplifying the controller's work.

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.

TxDone

Bit Value	(0011 1XXX XXXX XXXX)

The TxDone command is used by the external SMBus controller only. When the SMBus controller issues a TxDone command, it signals to the NIC that the data which has been downloaded to the transmit FIFO is a complete packet.

Issuing TxDone has the effect of "flushing" any remaining data in the host interface registers into the FIFO.

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.

TxFifoBisect

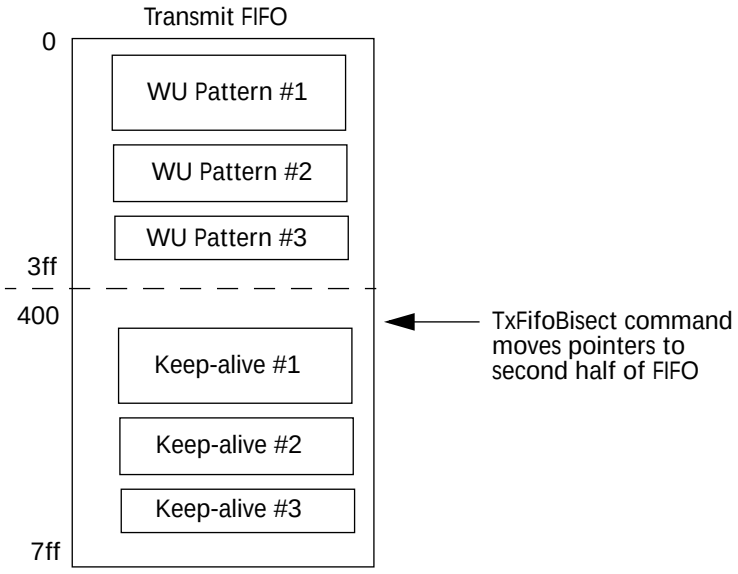
Bit Value	(1101 1XXX XXXX XXXX)

The TxFifoBisect command causes the transmit FIFO to be split logically into two 1K FIFOs. It should be used to prepare the transmit FIFO to accept keep-alive packets from the host.

The TxFifoBisect command causes the CurSopPtr, CurWritePtr, TopSopPtr, and TopReadPt registers to be pointed at the first word in the second half of the transmit FIFO RAM.

The purpose of splitting the FIFO is to allow both keep-alive packets (periodically sent based on a time interval) and pattern-matching packets (wake-up packets) to coexist in the transmit FIFO, as illustrated in Figure 11.

Figure 11 TxFifoBisect Command



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.

RxDiscard

Bit Value	(0101 0XXX XXXX XXXX)

The RxDiscard command is used by the external SMBus controller only. When the SMBus controller issues an RxDiscard command, it causes the top receive packet to be discarded.

An RxDiscard must be issued for every receive packet when removed through the SMBus interface. If the top packet has been completely read out of the receive FIFO, then RxDiscard causes the RxStatus register to reflect the status of the next packet in sequence. If the top packet is still being received or has only been partially read, RxDiscard causes the remainder of the packet to be discarded, and the data and status of the next packet to become available through the SmbFifoData and RxStatus registers, respectively.

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

Bit Value	(1100 1•XX XX•• ••••)

The SetHashFilterBit 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. Incoming packets that have the group bit set have a cyclic redundancy check (CRC) applied to their destination address.

The low-order six 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.

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
[1]	linkEventAck
[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

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

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 3C90xC NICs.

Overview

The auto-negotiation function on the 3C90xC NIC 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.

The auto-negotiation registers are different on the different versions of the ASIC that may be used on 3C90xC NICs. After you identify the ASIC version on the NIC, use the information in this chapter that applies to that version. The ASIC version numbers and ways to identify the ASICs are described in "ASICs" in Chapter 2.

40-0574-xxx or 40-05772-xxx ASIC Auto-Negotiation Registers

On a 3C90xC NIC that uses the 40-0574-xxx or 40-05772-xxx 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 NIC.

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 "Support for Signaling Standards" 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.

Table 18 summarizes the names, addresses, and functions of the 40-0574-xxx or 40-05772-xxx ASIC auto-negotiation registers. The registers are described in alphabetical order in the section following the table.

Table 18 Summary of 40-0574-xxx or 40-05772-xxx ASIC Auto-Negotiation Registers

Address	Register Name	Description
00	Control	Contains control bits to reset, restart, and configure auto-negotiation.
01	Status	Contains various status and capabilities bits.

Table 18 Summary of 40-0574-xxx or 40-05772-xxx ASIC Auto-Negotiation Registers

Address	Register Name	Description
02	PHY Identification 1	Contains PHY identification data.
03	PHY Identification 2	Contains PHY identification data.
04	Autonegotiation Advertisement	Controls which capabilities the NIC is allowed to advertise to the link partner.
05	Autonegotiation Link Partner Ability	Returns the advertised abilities received from the link partner during auto-negotiation.
06	Autonegotiation Expansion	Contains bits pertaining to expanded auto-negotiation functions.
07	Next Page Transmit	See the register bit definitions for a description of this register.
08-15	Reserved	
28	Device Specific 1	See the register bit definitions for a description of this register.
29	Device Specific 2	See the register bit definitions for a description of this register.
30	Device Specific 3	See the register bit definitions for a description of this register.
31	Quick Status	See the register bit definitions for a description of this register.

Autonegotiation Advertisement

The read/write Autonegotiation Advertisement register controls which capabilities the NIC is allowed to advertise to the link partner.

Autonegotiation Advertisement Register Format

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

Autonegotiation Advertisement Bit Descriptions

Bit	Name	Description
[4:0]	selector field	These read-only bits are hardwired with the value 00001 for IEEE 802.3.
[5]	10Base-T	When this bit is set, auto-negotiation advertises that the ASIC is capable of 10BASE-T operation.
[6]	10Base-T full duplex	When this bit is set, auto-negotiation advertises that the ASIC is capable of 10BASE-T full-duplex operation.
[7]	100Base-TX	When this bit is set, auto-negotiation advertises that the ASIC is capable of 100BASE-TX operation.
[8]	100Base-TX full duplex	When this bit is set, auto-negotiation advertises that the ASIC is capable of 100BASE-TX full-duplex operation.
[9]	100Base-T4	This bit is always set to a logic 0.
[10]	advertise pause capability	This bit is always set to on.
[13]	remote fault	When this read-only bit is set to a logic 1, the ASIC indicates to the link partner a remote fault condition.
[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. This function allows the exchange of arbitrary pieces of data. Data is carried by operational next pages of information.

Autonegotiation Expansion

The read-only Autonegotiation Expansion register Contains bits pertaining to expanded auto-negotiation functions.

Autonegotiation 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					

Autonegotiation Expansion Bit Descriptions

Bit	Name	Description
[0]	link partner autonegotiation capable	Setting this bit indicates that the link partner is capable of auto-negotiation.
[1]	page received	This read-only bit is set when a new link code word page has been received from the link partner. This bit is latched high (R/HL).
[2]	next page able	Setting this bit indicates that this device supports the next page function.
[3]	link partner next page able	Setting this bit indicates that the link partner supports the next page function.
[4]	parallel detection fault	Setting this bit indicates that a fault has been detected in the parallel detection function. This fault is due to more than one technology detecting concurrent link conditions. This bit can only be cleared by reading this register (R/LH).

Autonegotiation Link Partner Ability

The read-only Autonegotiation Link Partner Ability register Returns the advertised abilities received from the link partner during auto-negotiation.

Autonegotiation Link Partner Ability Register Format

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

Autonegotiation Link Partner Ability Bit Descriptions

Bit	Name	Description
[4:0]	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.
[12:5]	technology ability field	This field contains the technology ability of the link partner. These bits are defined as shown in the Autonegotiation Advertisement register bits [12:5].
[13]	remote fault	Setting this bit indicates that the link partner has a fault.
[14]	acknowledge	Setting this bit indicates that the link partner has successfully received at least three consecutive and consistent FLP bursts.
[15]	next page	Setting this bit indicates that the link partner wants to engage in next page exchange.

Control

The Control register contains control bits to reset, restart, and configure auto-negotiation.

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	When this bit is set to a logic 0, the ASIC asserts the COL signal in response to TXEN. This bit should only be set in loopback mode.
[8]	duplex mode	This bit reflects the mode of operation: ■ 1 = full-duplex ■ 0 = half-duplex This bit is ignored when the autonegotiation enable bit [12] is enabled. The default state is a logic 0. This bit is ORed with the TXLED[A] signal during power up and reset. This register is also ORed with the H_DUPLEX[D] signal during power up or reset.
[9]	restart autonegotiation	When this bit is set to a logic 1, the auto-negotiation process starts. The NwayDone bit in the StatusRegister is reset when this bit goes high. This bit is self-cleared when NWAY starts and its default is 0.
[10]	isolate	Setting this bit indicates that MII outputs should be in high impedance state. The default state is a logic 0.
[11]	powerdown	When this bit is set to a logic 1, the ASIC is placed in a low-power state. While in a powerdown state, the ASIC responds to management transactions. The default is 0.
[12]	autonegotiation enable	Wen this bit is set to a logic 1, the auto-negotiation process is enabled. The default is 0.
[13]	speed selection	The value of this bit reflects the current speed of operation: ■ 1 = 100 Mbps ■ 0 = 10 Mbps This bit only affects operating speed when the autonegotiation enable bit [12] is disabled. This bit is ignored when auto-negotiation is enabled. This bit is ANDed with the SPEEDLED[D] signal during power-up or reset.
[14]	loopback	When this bit is set, data transmission does not take 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 0.
[15]	reset	Setting this bit resets the ASIC. All registers are set to their default states. This bit is self-clearing. The default is 0.

Device Specific 1

The Device Specific 1 register is read-only.

Device Specific 1 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									

Device Specific 1 Bit Descriptions

Bit	Name	Description
[0]	link up 10	Setting this bit indicates that a 10 Mbps transceiver is up and operational.
[1]	link up 100	Setting this bit 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	Setting this bit indicates a false carrier indication. This bit latches high until it is read (R/LH).
[4]	unlocked	Setting this bit indicates TX scrambler lost lock. This bit latches high until it is read (R/LH).
[5]	disconnect	Setting this bit 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	This bit is a logic 1 when bit [3] of the Device Specific 3 Register is set. The autopolarity status bit indicates that the ASIC has detected and corrected a polarity reversal on the twisted-pair. This bit is not valid in 100 Mbps operation.
[7]	code violation	Setting this bit indicates that a Manchester code violation has occurred. The error code is outputted on the RXD lines. The the RXC signal descriptions for error code descriptions. This bit is valid in 10 Mbps mode only. This bit latches high and is only clear after it has been read or the device has been reset.
[8]	bad frame	Setting this bit indicates that a packet has been received without an SFD. This bit is valid in 10 Mbps mode only. This bit latches high and is only clear after it has been read or the device has been reset (R/LH).

Device Specific 2

The Device Specific 2 register is read/write.

Device Specific 2 Register Format

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

Device Specific 2 Bit Descriptions

Bit	Name	Description
[2]	jam enable	Setting this bit enables JAM associated with carrier integrity to be ORed with COL.
[3]	carrier integrity enable	Setting this bit enables carrier integrity.
[4]	scrambler/descrambler bypass	Setting this bit disables the scrambling and descrambling functions.
[5]	symbol aligner bypass	Setting this bit disables the aligner function.
[6]	encoder/decoder bypass	Setting this bit disables the 4B/5B encoder and the 5B/4B decoder functions.

Device Specific 2 Bit Descriptions (continued)

Bit	Name	Description
[8]	packet error indication enable	When this bit is set, a packet error code is reported on RXD[3:0] of the ASIC when RXER is asserted on the MII. A logic 0 disables this function. A packet error code indicates that the scrambler is locked.
[9]	link error indication	WHen this bit is set, a link error code is reported on RXD[3:0] of the ASIC when RXER is asserted on the MII. The specific error codes are listed int9he RXD signal description. A logic 0 disables this function.
[10]	carrier sense select	When this bit is set to a logic 1, CRS is asserted on receive only. When this bit is set to a logic 0, CRS is asserted on receive or transmit.
[12]	100 Mbps transmitter off	Setting this bit forces TXOP low and TXON high.
[13]	generic reset 1	
[14]	generic reset 2	
[15]	management reset	Setting this bit causes the lower 16 registers and registers 28 and 29 (Device Specific 1 and Device Specific 2) to be reset to their default values. This bit is self-clearing.

Device Specific 3

The Device Specific 3 register is read/write.

Device Specific 3 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						

Device Specific 3 Bit Descriptions

Bit	Name	Description
[0]	no lp mode	Setting this bit allows 10 Mbps operation with link pulses disabled. If the ASIC is configured for 10 Mbps operation, setting this bit does not affect operation.
[1]	serial select	Setting this bit selects 10 Mbps serial mode operation. If the ASIC is in 100 Mbps mode, this bit is ignored.
[2]	reference select	When this bit is set, the external 10 MHz reference signal REF10 is used for phase alignment.
[3]	autopolarity function enable	When this bit is set and the ASIC is in 10 Mbps mode, the autopolarity function determines if the TOP link is wired with a polarity reversal. If there is a polarity reversal, the ASIC asserts bit [6] of the Device Specific 1 Register and corrects the polarity reversal. If this bit is a logic 0 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 bit is valid in 10 Mbps mode only.
[5]	heartbeat enable	When this bit is set, the heartbeat function is enabled. This bit is valid in 10 Mbps mode only.

Next Page Transmit

The Next Page Transmit register is read/write.

Next Page Transmit Register Format

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

Next Page Transmit Bit Descriptions

Bit	Name	Description
[10:0]	message/unformatted code field	With these 11 bits, there are 2,048 possible messages. The 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. This bit always takes the opposite value of the toggle bit in e previously exchanged link code word. If the bit is a logic 0, the previous value of the transmitted link code word is a logic 1. If the bit is a logic 1, the previous value of the transmitted link code word is a logic 0. The initial value of the toggle bit in the first next page that is transmitted is the inverse of the value of bit [11] in the base link code word and therefore may assume a value of logic 1 or 0.
[12]	acknowledge 2	This bit is used by the next page function to indicate that a device has the ability to comply with the message. ■ Logic 0 = the device cannot comply with the message. ■ Logic 1 = the device complies with the message.
[13]	message page	This bit is used to differentiate a message page from an unformatted page. ■ Logic 0 = an unformatted page ■ Logic 1 = a formatted pate
[14]	acknowledge	This read-only bit is the acknowledge bit from the link code word.
[15]	next page	This bit indicates whether this is the last next page to be transmitted. ■ Logic 0 = this is the last page ■ Logic 1 = there is an additional next page

PHY Identification 1

The read-only PHY Identification 1 register contains PHY identification data.

PHY Identification 1 Register Format

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

PHY Identification 1 Bit Descriptions

Bit	Name	Description
[0:15]	organizationally unique identifier	Bits [3:24]of the organizationally unique identifier (OUI) that is assigned to the PHY manufacturer by the IEEEE are placed in this field and in MR3 PHY Identification 2 register [0:15].

PHY Identification 2

The read-only PHY Identification 2 register contains PHY identification data.

PHY Identification 2 Register Format

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

PHY Identification 2 Bit Descriptions

Bit	Name	Description
[3:0]	revision number	These bits are the value of the current revision number (01h for the first revision.
[9:4]	model number	The six-bit model number of the device. The model number is 24h.
[15:10]	organizationally unique identifier	Bits [3:24]of the organizationally unique identifier (OUI) that is assigned to the PHY manufacturer by the IEEEE are placed in the PHY Identification 1 register [0:15] and in this field.

Quick Status

The Quick Status register is read-only.

Quick Status Register Format

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

Quick Status Register Bit Descriptions

Bit	Name	Description
[2:0]	highest autonegotiation state	These bits report the highest auto-negotiation state reached since the last register read. They are defined in the following priority order: ■ 000 = auto-negotiation enabled ■ 001 = transmit disable or ability detect ■ 010 = link status check ■ 011 = acknowledge detect ■ 100 = complete acknowledge ■ 101 = FLP link good check ■ 110 = next page wait ■ 111 = FLP link good
[5:3]	lowest autonegotiation state	These bits report the lowest auto-negotiation state reached since the last register read, as defined in bit [2:0].
[8]	duplex mode	This bit, when set to a logic 1, indicates that the link is operating at full-duplex mode. A logic 0 indicates that the link is operating in half-duplex mode.

Quick Status Register Bit Descriptions (continued)

Bit	Name	Description
[9]	link speed	This bit, when set to a logic 1, indicates that the link is operating at 100 Mbps. A logic 0 indicates that the link is at 10 Mbps.
[10]	link partner pause	This bit, when set to a logic 1, indicates that both link partners have negotiated to exchange pause information.
[11]	link status	Setting this bit indicates that a valid link is established. This bit has a latching low function. A link failure causes the bit to clear and stay cleared until it has been read through the management interface.
[12]	unlocked/jabber	Setting this bit in 100 Mbps mode indicates that the TX scrambler has lost lock. In 10 Mbps mode, this bit indicates that a jabber condition has been detected. This bit remains set until it is cleared by reading the register.
[13]	remote fault	Setting this bit indicates that a remote fault has been detected. This bit remains set until it is cleared by reading the register. The default is a logic 0.
[14]	false carrier	Setting this bit indicates that the carrier detect state machine has found a false carrier. This bit remains set until it is cleared by reading the register. The default is a logic 0.
[15]	receive error	Setting this bit indicates that a receive error has been detected. This bit is valid in 100 Mbps mode only. This bit remains set until it is cleared by reading the register. The default is a logic 0.

Status The read-only 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, indicating that the ASIC supports the extended register set (PHY Identifier 1 and beyond).
[1]	jabber detect	This bit is set whenever a jabber condition is detected. This bit remains set until it is read and the jabber condition no longer exists.
[2]	link status	When this bit is set to a logic 1, a valid link has been established. 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.

MR1: Status Bit Descriptions (continued)

Bit	Name	Description
[4]	remote fault	Setting this bit indicates that a remote fault has been detected. This bit remains set until it is cleared with a read. The default is a logic 0.
[5]	autonegotiation complete	When this bit is set to a logic 1, the auto-negotiation process is complete and the contents of the following registers are valid: ■ MR4: autonegotiation advertisement ■ MR5: autonegotiation link partner ability ■ MR6: autonegotiation expansion ■ MR7: next page transmit This bit is reset when auto-negotiation starts. The default is a logic 0.
[6]	suppress preamble	This bit is always set to indicate that the ASIC accepts management frames with the preamble suppressed.
[11]	10Base-T half duplex ability	This bit always returns a logic 1, indicating auto-negotiation's ability to negotiate a 10BASE-T half-duplex link.
[12]	10Base-T full duplex ability	This bit always returns a logic 1, indicating auto-negotiation's ability to negotiate a 10BASE-T full-duplex link.
[13]	100Base-TX half duplex ability	This bit always returns a logic 1, indicating auto-negotiation's ability to negotiate a 100BASE-TX half-duplex link.
[14]	100Base-TX full duplex ability	This bit always returns a logic 1, indicating auto-negotiation's ability to negotiate a 100BASE-TX full-duplex link.
[15]	100Base-T4 ability	This bit is always set to a logic 0.

(2 of 2)

40-0579-xxx ASIC Auto-Negotiation Registers

The 40-0579-xxx ASIC has an integrated 802.3u auto-negotiation function that handles auto-negotiation for 10BASE-T and 100BASE-TX media types. 100BASE-T4 is not implemented on the 3C90xC NIC.

The 40-0579-xxx 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 40-0579-xxx 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.

Table 19 summarizes the names, addresses, and functions of the 40-0579-xxx ASIC auto-negotiation and MII registers. The registers are described in alphabetical order in the sections following the table.

Table 19 Summary of 40-0579-xxx 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	Autonegotiation 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	Autonegotiation 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	Reserved	
16h	Reserved	
17h	PTEST	Reserved.
18h	Auxiliary Status Summary	See the register bit definitions for a description of this register.
1Ah	INTERRUPT	See the register bit definitions for a description of this register.
1Bh	Reserved	
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.
1Fh	BROADCOM Test	Reserved.

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 PHY core, 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 indicates whether or not full-duplex mode is active. ■ 0 = full-duplex not active ■ 1 = full-duplex active
[1]	speed indication	This read-only bit indicates the current operation speed of the NIC. ■ 0 = 10BASE-T operation ■ 1 = 100BASE-X operation 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 zero when one of the following instances is true: ■ The ANEN pin is low and the F100 pin is low. ■ The autonegotiation enable bit [12] and the forced speed selection bit [13] in the Control register are set to zero. When bit [8] of the Auxiliary Control register is zero, the speed of the chip is 10BASE-T. In all other cases, either the speed is not forced (auto-negotiation is enabled), or the speed is forced to 100BASE-X. ■ 0 = speed forced to 10BASE-T ■ 1 = speed forced to 100BASE-X
[3]	auto-negotiation indication	This bit indicates whether auto-negotiation has been enabled or disabled on the ASIC. A combination of a 1 in the autonegotiation enable bit [12] in the Control register and a logic 1 on the ANEN input pin is required to enable auto-negotiation. When auto-negotiation is disabled, bit [15] of the Auxiliary mode register returns a 0. At all other times, it returns a 1. ■ 0 = speed is manually forced ■ 1 = auto-negotiation activated
[4]	repeater mode indication	This bit returns the same value as the RPTR input line. ■ 0 = the ASIC is in DTE mode ■ 1 = the ASIC is in repeater mode
[5:7]	revision	These read-only bits return the revision number of the ASIC. The current revision is labeled 001.
[8]	polarity error	This bit reflects the polarity status of the receive channel pair. The ASIC 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. Instead, this bit returns a 1 whenever the polarity of the channel is inverted. ■ 0 = channel polarity correct ■ 1 = channel polarity inverted

10BASE-T Auxiliary Error and General Status Bit Descriptions (continued)

Bit	Name	Description
[9]	EOF error	This bit returns a one when the end of frame (EOF) sequence is improperly received. 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. This bit is valid during 10BASE-T operation only.

(2 of 2)

100BASE-X Auxiliary Control

The 100BASE-X Auxiliary Control register is read/write.

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		

100BASE-X Auxiliary Control Bit Descriptions

Bit	Name	Description
[1]	MII out-of-band enable	This bit controls the MII out-of-band mechanism within the MII receive logic. ■ 0 = disabled ■ 1 = enabled
[2]	extended MII FIFO enabled	This bit controls the extended MII FIFO mechanism. ■ 0 = disabled ■ 1 = enabled
[5]	FEF enable	This bit controls the Far End Fault (FEF) mechanism that is associated with 100BASE-FX operation. ■ 0 = disabled ■ 1 = enabled
[6]	baseline wander correction disable	This bit, when set to 1, 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. 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.
[8]	bypass MLT3 encoder/decoder	This bit, when set to one, bypasses the MLT3 encoder and decoder. NRZ 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.

(1 of 2)

100BASE-X Auxiliary Control Bit Descriptions (continued)

Bit	Name	Description
[10]	bypass 4B5B encoder/decoder	This bit, when set to one, bypasses the 4B5B encoder and decoder. The transmitter sends 5B codes from the TXER and TXD [3:0] pins directly to the scrambler. TXEN must be active and frame encapsulation (insertion of J/K and T/R codes) are not performed. 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.
[12]	CIM disable	This bit, when set to 1, disables the carrier integrity monitor for this port. When set to 0, the CIM function is enabled. The default value of this bit after reset is determined by the CIM mode select input , if RPTR=1. If RPTR=0 during reset, the CIM function is disabled by default.
[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											

100BASE-X Auxiliary Status Bit Descriptions

Bit	Name	Description
[0]	MLT3 code error detected	This bit returns a 1 if an MLT3 coding error is detected in the receive data stream since the last time this register was read; otherwise, it returns a 0.
[1]	lock error detected	This bit returns a 1 if the descrambler has lost lock since the last time this register was read; otherwise, it returns a 0.
[2]	transmit error detected	This bit returns a 1 if a packet is received with a transmit error code since the last time this register was read; otherwise, it returns a 0.
[3]	receive error detected	This bit returns a 1 if a packet was received with an invalid code since the last time this register was read; otherwise, it returns a 0.
[4]	bad ESD detected	This bit returns a 1 if a bad end of stream error has been detected since the last time the register was read; otherwise, it returns a 0.
[5]	false carrier detected	This bit returns a 1 if a false carrier is detected since the last time the register was read; otherwise, it returns a zero.
[6]	disconnect state	This bit returns a 1 when the link is unstable and the carrier integrity monitor has isolated the port; otherwise, it returns a 0. This bit is qualified by 100BASE-X operation.

(1 of 2)

100BASE-X Auxiliary Status Bit Descriptions (continued)

Bit	Name	Description
[7]	remote fault	This bit returns a 1 while its link partner is signaling a far-end fault condition; otherwise, it returns a 0.
[8]	current 100BASE-X link status	This bit returns a 1 when the 100BASE-X link status is good; otherwise, it returns a 0.
[9]	locked	This bit returns a 1 when the descrambler is locked to the incoming data stream; otherwise, it returns a zero.
[10]	FX mode	This bit returns a 1 when SD± input pins are driven with a valid differential signal level. It returns a 0 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.

When the counter reaches its maximum value, FFh, it stops counting disconnects until cleared.

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.

When the counter reaches it maximum value (FFh), it stops counting false carrier sense errors until cleared.

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.

When the counter reaches its maximum value (FFFFh), it stops counting receive errors until cleared.

100BASE-X Receive Error Counter Register Format

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

Auto-Negotiation Advertise Bit Descriptions

Bit	Name	Description
[4:0]	advertise selector field	These read-only 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. The default value reflects the abilities of the NIC. Resetting the NIC restores the default value. 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. The default value reflects the abilities of the NIC. Resetting the NIC restores the default value. 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. The default value reflects the abilities of the NIC. Resetting the NIC restores the default value. 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).

(1 of 2)

Auto-Negotiation Advertise Bit Descriptions (continued)

Bit	Name	Description
[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. The default value reflects the abilities of the NIC. Resetting the NIC restores the default value. 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).
[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. The default value reflects the abilities of the NIC. Resetting the NIC restores the default value. 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]	advertise pause operation	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.
[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. 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.
[15]	next page	This bit must always be written 0.

(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					

Auto-Negotiation Expansion Bit Descriptions

Bit	Name	Description
[0]	link partner auto-negotiation able	This read-only 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.
[1]	page received	This read-only 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.

Auto-Negotiation Expansion Bit Descriptions (continued)

Bit	Name	Description
[2]	next page able	This read-only bit always returns 0.
[3]	link partner next page able	This read-only 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. 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	0								

Auxiliary Control/Status Bit Descriptions

Bit	Name	Description
[0]	full-duplex indication	This read-only 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. 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: ■ The ANEN pin is low and the F100 pin is low. ■ The auto-negotiation enable bit (12) of the Control register is set to zero. 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. 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. When auto-negotiation is disabled, this bit returns a 0.

(1 of 2)

Auxiliary Control/Status Bit Descriptions (continued)

Bit	Name	Description
[4:5]	edge rate	These control bits are used to program the transmit DAC output edge rate in 100BASE-TX mode. 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). ■ 00 = 1 ns ■ 01 = 2 ns ■ 10 = 3 ns ■ 11 = 4 ns
[6:7]	HSQ:LSQ	These bits extend or decrease the squelch levels for detection of incoming 10BASE-T data packets. 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. 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.
[14]	link disable	This bit, when set to 1, disables the link integrity state machines and places the NIC into forced link pass status. When set to 0, this bit restores the link integrity functions. Resetting the chip also restores the link integrity functions. Reading this bit returns the value of the link integrity disable status.
[15]	jabber disable	This bit, when set to 1, disables the jabber detect function, as defined in the IEEE standard. This function shuts off the transmitter when a transmission request has exceeded a maximum time limit. When this bit is set to 0, or when the chip is reset, normal operation is enabled. 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	0	0	0			0		0

Auxiliary Mode Bit Descriptions

Bit	Name	Description
[1]	block TXEN mode	This bit, when set to 1, enables the block TXEN mode. When this mode is enabled, short IPGs of one, two, three, or four TXC cycles all result in the insertion of two IDLEs before the beginning of the next packet's JK symbols.
[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.
[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.

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		

Auxiliary Multiple PHY Bit Descriptions

Bit	Name	Description
[0]	RXER code mode	■ 0 = disable RXER code mode ■ 1 = enable RXER code mode
[1]	10Base-T serial mode	■ 0 = disable 10BASE-T serial mode ■ 1 = enable 10BASE-T serial mode
[3]	super isolate	This bit, when set to one, places the NIC in super isolate mode. When set to zero, it enables normal operation. 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. 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.
[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. It remains this value until the auto-negotiation process is restarted, a link fault occurs, auto-negotiation is disabled, or the NIC is reset.

(1 of 2)

Auxiliary Multiple PHY Bit Descriptions (continued)

Bit	Name	Description
[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. 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. 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. 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)

Auxiliary Status Summary

The Auxiliary Status Summary register contains redundant status bits found elsewhere within the MII register space.

Auxiliary Status Summary Register Format

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

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. This bit is implemented in 10BASE-T operation only.

(1 of 3)

Auxiliary Status Summary Bit Descriptions (continued)

Bit	Name	Description
[1]	auto-negotiation enabled	Auto-negotiation can be disabled by one of two methods: hardware or software control. 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. When auto-negotiation is disabled by software control, writing a one to this bit or resetting the NIC enables auto-negotiation. 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. 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. 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.
[8:10]	auto-negotiation HCD	This bit indicates the highest common denominator (HCD) discovered with auto-negotiation. ■ 000 = No highest common denominator ■ 001 = 10BASE-T ■ 010 = 10BASE-T full-duplex ■ 011 = 100BASE-TX ■ 100 = 100BASE-T4 ■ 101 = 100BASE-TX full-duplex ■ 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.

Auxiliary Status Summary Bit Descriptions (continued)

Bit	Name	Description
[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.

(3 of 3)

Control

The Control register contains control bits to reset, restart, and configure auto-negotiation.

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. 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.
[8]	duplex mode	When this bit is set to one and the autonegotiation 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. 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. 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 [9] in the Auxiliary Multiple PHY register.

(1 of 2)

Control Bit Descriptions (continued)

Bit	Name	Description
[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. 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. 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]	autonegotiation enable	Auto-negotiation can be disabled by one of two methods: hardware or software control. 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. When auto-negotiation is disabled by software control, writing a one to this bit or resetting the chip enables auto-negotiation. 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. 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. When this bit is read, it returns the value of the software-controlled forced speed selection only.
[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. 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. This bit clears itself after the reset process is complete and does not need to be cleared using a second MII write. 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.

(2 of 2)

Interrupt

Interrupt Register Format

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

Interrupt Bit Descriptions

Bit	Name	Description
[0]	INTR status	This read-only bit represents the status of the INTR# input. A 1 indicates that the interrupt mask is off and that one or more of the change bits are set. Reading this register clears this bit.
[1]	LINK change	This read-only bit, when set to 1, indicates a change of link status since the last register read. Reading this register clears this bit.
[2]	SPD change	This read-only bit, when set to 1, indicates a change of speed status since the last register read. Reading this register clears this bit.
[3]	FDX change	This read-only bit, when set to 1, indicates a change of duplex status since the last register read. Reading this register clears this bit.
[8]	INTR mask	When this bit is set, no interrupts are generated, regardless of the state of the other MASK bits.
[9]	LINK mask	When this bit is set, changes in link status do not generate an interrupt.
[10]	SPD mask	When this bit is set, changes in operating speed do not generate an interrupt.
[11]	FDX mask	WHen this bit is set, changes in duplex mode do not generate an interrupt.
[14]	INTR enable	This bit, when set, enables interrupt mode. This bit and the FDX LED enable bit [15] are exclusive; only one may be set at a time. When interrupt mode is enabled, XMTLED# becomes INTR# and RCVLED# becomes ACTLED#.
[15]	FDX LED enable	This bit, when set, enables full-duplex LED mode. This bit and the INTR enable bit [14] are exclusive; only one may be set at a time.

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											

Link Partner Ability Bit Descriptions

Bit	Name	Description
[0:4]	link partner selector field	These read-only 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.

(1 of 2)

Link Partner Ability Bit Descriptions (continued)

Bit	Name	Description
[5]	LP advertise 10BASE-T	This read-only 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 read-only 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 read-only 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 read-only 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-T4	This read-only 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.
[10]	LP advertise pause	This read-only bit, when set to one, indicates that the link partner is capable of pause operation.
[13]	LP remote fault	This read-only 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 read-only bit indicates that a device has successfully received the link partner's link code word.
[15]	LP next page	This read-only bit is the link partner next page bit.

(2 of 2)

PHYID High

The PHYID High register contains PHY identification data (also known as OUI—organizationally unique identifier—data).

PHYID High Register Format

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

PHYID High Bit Descriptions

Bit	Name	Description
[0:15]	Address 00010:ID MSBs	These are the most-significant bits of the PHY. PHYID High [15:0] = OUI [21:6] (OUI bits [23:22] are not represented.)

PHYID Low

The PHYID Low register contains PHY identification data (also known as OUI—organizationally unique identifier—data).

PHYID Low Register Format

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

PHYID Low Bit Descriptions

Bit	Name	Description
[0:15]	Address 00011:ID LSBs	These are the 16 least-significant bits of the PHY. PHYID Low [15:10] = OUI [5:0]). These bits are the OUI least-significant bits. PHYID Low [9:4] (Model [5:0]). These bits are the vendor model number. PHYID Low [3:0] (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							

Status Bit Descriptions

Bit	Name	Description
[0]	extended capability	The NIC supports extended capability registers and returns a one when this read-only 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. This bit is implemented in 10BASE-T operation only.
[2]	link status	This read-only 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. 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 read-only bit returns a one when read, regardless of whether or not the auto-negotiation function has been disabled.
[4]	remote fault	This read-only 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. 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. The bit returns a zero if the auto-negotiation process is not complete.

Status Bit Descriptions (continued)

Bit	Name	Description
[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. 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. When this bit is set to zero, a preamble is always required.
[11]	10BASE-T capability	This read-only bit returns a one when read, indicating that the NIC is capable of 10BASE-T half-duplex operation.
[12]	10BASE-T FDX capability	This read-only bit returns a one when read, indicating that the NIC is capable of 10BASE-T full-duplex operation.
[13]	100BASE-TX capability	This read-only bit returns a one when read, indicating that the NIC is capable of 100BASE-TX half-duplex operation.
[14]	100BASE-TX FDX capability	This read-only bit returns a one when read, indicating that the NIC is capable of 100BASE-TX full-duplex operation.
[15]	100BASE-T4 capability	This read-only bit always returns a 0.

12 OTHER REGISTERS

This chapter describes the following registers:

- BiosRomAddr
- BiosRomData
- ConfigAddress
- ConfigData
- DebugControl
- DebugData
- FifoDiagnostic
- Media
- NetworkDiagnostic
- PhysicalMgmt
- PowerMgmtCtrl
- ResetOptions
- SosBits
- 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

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

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

7	6	5	4	3	2	1	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.

ConfigAddress

Synopsis	Provides an I/O back door into the PCI configuration space..
Type	Read/write
Size	8 bits
Offset	44

ConfigAddress, in conjuction with the ConfigData register, provides a method to access the PCI configuration space through I/O accesses. This register can be written to a value of 0 to 256 directly. A subsequent read from the ConfigData register results in the data from that location. Similarily, a write to the ConfigData register writes that data into the PCI configuration location addressed by ConfigAddress.

ConfigAddress Register Format

7	6	5	4	3	2	1	0

ConfigData

Synopsis	PCI configuration data register.
Type	Read/write
Size	8 bits
Offset	48

The ConfigData registeris an 8-bit, read/write data register that is used to access the PCI configuration location addressed by the ConfigAddress register.

DebugControl

Synopsis	Diagnostics control register.
Type	Read/write
Size	16 bits
Offset	74

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. 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

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.

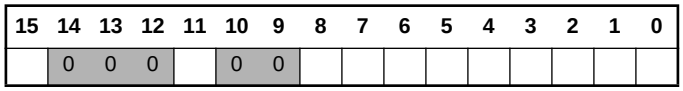
FifoDiagnostic

Synopsis	Provides diagnostic read access to the packet-buffering (FIFO) logic.
Type	Read/write
Size	16 bits
Window	4
Offset	4

The bits in the FifoDiagnostic register provide various indications of transmit and receive FIFO failures.

Bits [8:0] 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.

FifoDiagnostic Register Format



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.
[8]	smbPMEMask	This read/write bit, when set, blocks a PME event from going onto the PCI bus. The default is 0.
[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. 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. This bit does not in itself indicate an overrun condition. However, if data is received while this bit is set, an overrun will occur. This bit is informational only. This bit is cleared as soon as the receive FIFO is no longer full.

(1 of 2)

FifoDiagnostic Bit Descriptions (continued)

Bit	Name	Description
[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.

(2 of 2)

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											

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. 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: ■ 0 = Standard IFS + 0 bit times ■ 1 = Standard IFS + 16 bit times ■ 2 = Standard IFS + 32 bit times ■ 3 = Standard IFS + 64 bit times ■ 4 = Standard IFS + 96 bit times ■ 5 = Standard IFS + 128 bit times ■ 6 = Standard IFS + 160 bit times ■ 7 = Standard IFS + 192 bit times ■ 8 = Standard IFS + 224 bit times ■ 9 = Standard IFS + 256 bit times

(1 of 3)

MacControl Bit Descriptions (continued)

Bit	Name	Description
		■ A = Standard IFS + 288 bit times ■ B = Standard IFS + 320 bit times ■ C = Standard IFS + 352 bit times ■ D = Standard IFS + 384 bit times ■ E = Standard IFS + 416 bit times ■ F = Standard IFS + 448 bit times 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. 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. Software must issue TxReset and RxReset commands after changing the value of this bit. For information on programming fullDuplexEnable, see "Setting the Duplex Mode" in Chapter 4.
[6]	allowLargePackets	This bit determines the packet size at which the oversizedFrame error is generated for receive packets. The minimum packet sizes at which an oversizedFrame error will be flagged are: ■ allowLargePackets value 0 = 1519 minimum ■ allowLargePackets value 1 = 4495* minimum *This value was calculated by taking the maximum FDDI frame size, 4500 bytes, and subtracting bytes for fields that have no Ethernet equivalent. The packet size includes the destination and source addresses, the type/length field, and the FCS field.
[7]	extendAfterCollision	This bit determines the extended deference mode. ■ 0 = "Old-style" extended deference — Deference extension occurs after all transmissions, including collisions. ■ 1 = Deference extension occurs only after a collision, and only when this station was the last to transmit a packet.
[8]	flowControlEnable	Flow control enable. ■ 0 = Default. Treat all incoming packets normally. ■ 1 = Flow control enabled. Act upon incoming flow control (PAUSE) packets. Note: This bit should not be set unless fullDuplexEnable is also set.

MacControl Bit Descriptions (continued)

Bit	Name	Description
[9]	vltEnable	3Com-proprietary VLAN tagging (VLT) enable. ■ 0 = Default. Treat incoming packets as normal IEEE 802.3 frame format. ■ 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.
[10]	vlanOversizeEn	This bit, when set, allows IEEE 802.1Q frames to not be flagged as an oversizeFrame in the UpPacketStatus register if its packet size is less than or equal to the value in the MaxPktSize register, plus four. ■ 0 = Default. Any frame with a size greater than the value in the MaxPktSize register is flagged as an oversizedFrame. ■ 1 = IEEE 802.1Q frame with a size greater than the value in the MaxPktSize register, plus 4, is flagged as an oversizedFrame, and non-802.1Q frames with a size greater than the value in the MaxPktSize register are flagged as an oversizedFrame.

(3 of 3)

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 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 3C90xC NIC. 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. This bit is not implemented on 3C90xC NIC. It is programmed to zero.

(1 of 2)

MediaOptions Bit Descriptions (continued)

Bit	Name	Description
[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. 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.
[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.
[4]	coaxAvailable	This read-only bit, when set, indicates that a 10BASE2 coaxial transceiver is available on the NIC. This bit is not implemented on the 3C90xC NIC. 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. This bit is not implemented on the 3C90xC NIC. 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.
[8]	10BaseFL	This read-only bit, when set, indicates that a 10BASE-FL transceiver is available on the NIC. This bit is not implemented on the 3C90xC NIC. 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	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. 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.
[6]	jabberGuardEnable	This bit is for use only with the 10 Mbps twisted-pair transceiver. 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. When the NIC is operating in 10BASE-T mode, this bit reflects the state of the link beat logic. 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. 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.

(1 of 2)

MediaStatus Bit Descriptions (continued)

Bit	Name	Description
[14]	dcConverterEnabled	This bit, when set, indicates that the 10BASE2 DC-DC converter has been enabled with the EnableDcConverter command. This bit is not implemented on the 3C90xC NIC. It is programmed to zero.
[15]	auiDisable	This read-only bit is asserted whenever any media port except AUI has been selected. This bit always reads as a one on the 3C90xC NIC.

(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

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.

NetworkDiagnostic Register Format

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

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. This field is equivalent to bits [6:2] in the RevisionId PCI configuration register.

(1 of 2)

NetworkDiagnostic Bit Descriptions (continued)

Bit	Name	Description
[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. 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. 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. 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.
[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. 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	This bit, when set, enables PHY loopback.
[15]	externalLoopback	Setting this bit enables reception of packets transmitted by the NIC. Address-filtering criteria must also be met for each packet received. This is an external loopback and requires a loopback plug.

(2 of 2)

Table 20 summarizes the various loopback modes and the required values for the associated bits in the NetworkDiagnostic, MacControl, and PhysicalMgmt registers.

Table 20 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. This bit is implemented in the 3C90xC NIC but has no function.

PowerMgmtCtrl

Synopsis	Write-only version of the PowerMgmtCtrl PCI configuration regsiter.
Type	Write-only
Size	16 bits
Offset	7c

The PowerMgmtCtrl register is a write-only version of the PowerMgmtCtrl PCI configuration register. This version helps work around a problem that is found in pre-ACPI environments. In such systems, the driver (not the operating system) is required to change the power state by writing to the PowerMgmtCtrl register. However, the driver is prevented from performing configuration writes by Windows operating systems when the NIC is in the D0 power state. Mapping the register here allows the driver to lower the power state.

See the PowerMgmtCtrl register definition in Chapter 3 for a bit description and more details about accessing this register in I/O memory space.

ResetOptions

Synopsis	Provides read access to various configuration and test mode bits.
Type	Read/write (some bits read-only)
Size	16 bits
Window	2
Offset	c

The ResetOptions register contains bits that indicate the configuration and test status of the NIC.

ResetOptions Register Format

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

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. ■ 000 = Motherboard feature set: All advanced features are enabled except packet scheduling. ■ 001 = Low-cost NIC feature set: All advanced features are disabled. ■ 010 = Standard NIC feature set: All advanced features are enabled. ■ 100 = Server NIC feature set: All advanced features are enabled. Other combinations are reserved.
[3]	smBusDisable	This read/write bit, when set, disables the SMBus interface. The NIC does not respond to any incoming SMBus cycles.
[4]	smBusMode	When this read/write bit is set, it indicates that the NIC is being used in an SMBus signal-level-compatible application and drives this signal accordingly. If this bit is not set, the NIC drives I^2C signal levels on the SMBus clock and data lines.
[5]	disableAdv100	When this read/write bit is set, the auto-negotiation function is prevented from advertising 100 Mbps capability to its link partner. This bit is not implemented in the 3C90xC NIC. It always returns zero.
[6]	ee16KInstalled	This read/write bit, when set, indicates that a 16K-bit EEPROM device is installed on the NIC. If this bit is not set, the hardware assumes a 2K or 4K EEPROM device.
[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. This bit is not implemented in the 3C90xC NIC.
[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]	fastReset	This read-only bit, when set, places the NIC into a fast reset mode. This is intended for simulations only.
[12]	test100Tx	When this read-only bit is set, the ASIC is in a PDT bypass test mode. This bit is not implemented in the 3C90xC NIC. It always returns zero.

ResetOptions Bit Descriptions (continued)

Bit	Name	Description
[13]	test100Rx	When this read-only bit is set, the ASIC is in PDR bypass test mode. This bit is not implemented in the 3C90xC NIC. It always returns zero.
(2 of 2)		

SosBits

Synopsis	Reflects the state of the external SOS pins.
Type	Read-only
Size	7 bits
Window	1
Offset	a

The SosBits register shows the state of the NIC's external SOS pins. This register is only valid when the device is in the 10BASE-T, 100BASE-TX, or auto-negotiation mode.

SosBits Register Format

7	6	5	4	3	2	1	0
0							

Bits [0:6] correspond to the SOS pins 1 to 7, respectively.

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

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

The Countdown register is a programmable down-counter that can cause the NIC to generate an interrupt when the counter expires. It is cleared by a reset.

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.

FreeTimer

Synopsis	Provides a free-running counter for general timing purposes.
Type	Read-only
Size	16 bits
Offset	34

FreeTimer Register Format

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

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 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

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.

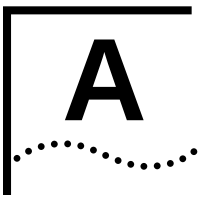
To use Timer for general-purpose measurements at driver initialization time, ensure that the interruptLatch bit in the IntStatus register is clear (a pending interrupt would prevent the counter from starting), disable system interrupts, and issue a RequestInterrupt command to start the timer.

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 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.

The on-chip auto-negotiation registers appear as a PHY device and are accessible through the management interface. A 3C90xC 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 3C90xC NIC. 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 3C90xC NIC are described in the sections following the table.

Table 21 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.
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.

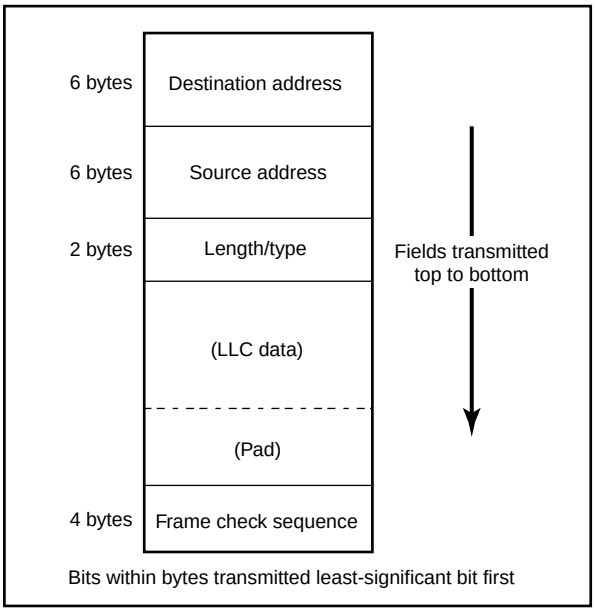
C

FRAME FORMATS

This appendix illustrates the frame formats.

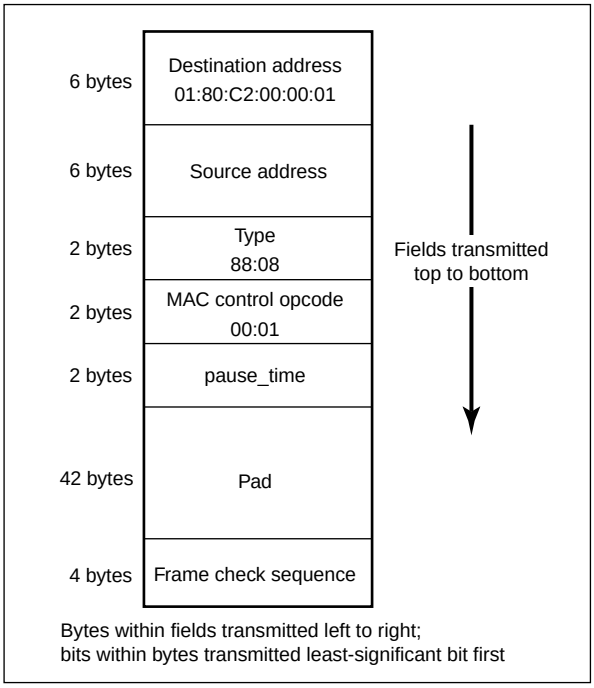
IEEE 802.3 MAC Frame Format

Figure 12 IEEE 802.3 MAC Frame Format



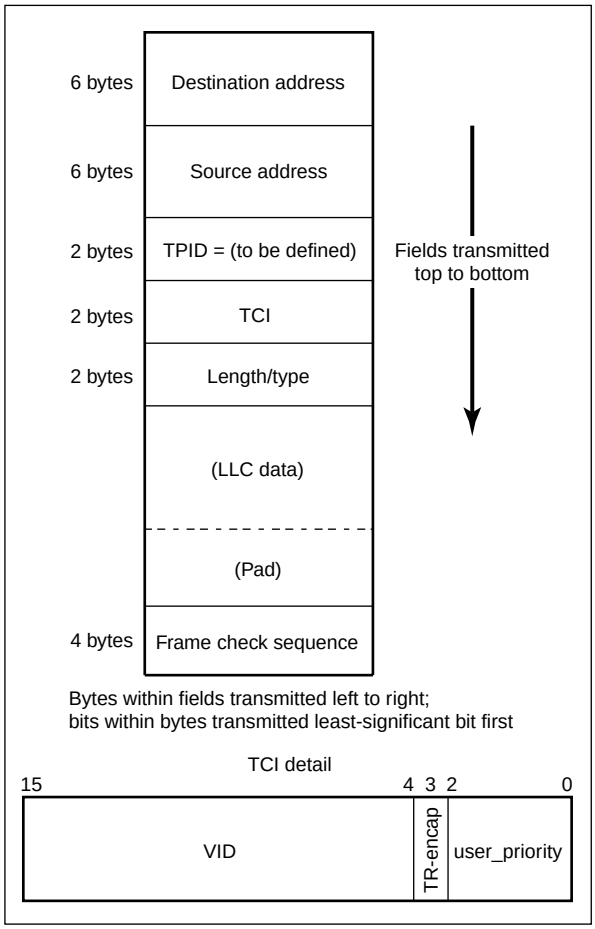
IEEE 802.3x PAUSE Frame Format

Figure 13 IEEE 802.3x PAUSE Frame Format



IEEE 802.1q Frame Format

Figure 14 IEEE 802.1q Frame Format



D

ERRATA LIST AND SOLUTIONS

This appendix describes 3C90xC NIC anomalies and possible solutions.

Table 22 3C90xC NIC Anomalies

ASIC(s)	Anomaly	Description	Solution
-003 V1	The transmission of KeepAlive and SOS packets becomes non-functional after AC power is restored from a complete power loss.	After restoration of AC power due to a complete power loss, such as the removal of the AC plug or power outage, the transmission of KeepAlive and SOS packets is interrupted. There are two cases in which this happens: ■ Case #1: In addition to restoration of axiliary power after AC power is restored, some systems have a momentary window in which there is +5Vpci, as well as a PciClk and RSTN signal. Note that the PciClk goes away when the PCI voltage falls approximately below +3.8V. The result of +5V on the PCI-bus causes the -003 V1 ASIC to cycle through different power modes. As AC power ramps up, the power-on reset of the low-voltage detect cell resets the ASIC to a D0 powerstate and full-PCI-powermode, and then kicks off the EEPROM loading. When the PCI voltage drops below +1.7V to trip the low-volt-age detect cell, the ASIC then switches to full-AUX-power mode and attempts to switch from the PciClk to the on-board 25MHz crystal oscillator. However, because the PciClk disappears well before the trip point of the low-voltage detect cell, the internal PciClk is unable to switch to the 25MHz crystal oscillator. This lack of a PciClk to the TxFIFO logic of the ASIC interrupts the transmission of KeepAlive or SOS packets. ■ Case #2: After AC power is restored, only auxiliary power is restored. When the ASIC powers up again with only auxiliary power, some logic is not cleared and left in unknown states. This can interrupt the transmission of KeepAlive and SOS packets.	■ Hardware Fix for Case #1: The new -004 V1 ASIC has a logic fix for Case#1. This logic fix involves clearing the flip-flop associated with switching the clock domain to either the PciClk or on-board 25MHz crystal oscillator. This flip-flop will be cleared when the PCI voltage drops below the threshold voltage, thus, switching the internal PCI domain to the on-board 25MHz crystal oscillator. ■ Software Fix for Case #1: If an ASIC roll is not an option for OEM customers, it is possible do a software workaround. The ASIC can be placed into full-AUX-powermode much earlier, when the PciClk is still available. This is accomplished by using the FlexEE portion of the EEPROM to put it in a non-D0 powerstate. A non-D0 state generates a signal that will put the ASIC into full-AUX-powermode and switch the internal clock from the PciClk to the on-board 25MHz crystal oscillator. This internal switching will happen well before the PciClk goes away due to power being removed. The only disadvantage to this is that the ASIC is then limited to only configuration I/O cycles since it is no longer in a D0 powerstate. However, it is possible to use the system BIOS to set the ASIC back to a D0 powerstate upon normal bootup. Note: Since the FlexEE portion of the EEPROM will put the ASIC in a D3 powerstate, then it is no longer valid to set bit7 autoResetToD0 in the SoftwareInformation2 (offset 0x0F) of the EEPROM. ■ No fix for Case #2 is available at present.

Table 22 3C90xC NIC Anomalies (continued)

ASIC(s)	Anomaly	Description	Solution
3C920V1-005	In certain systems which utilize the Intel 450NX PCI chipset, using two 3C905C NICs with ASIC revisions up to and including -005 may cause Microsoft Windows HCT Certification Tests to "blue screen" after 30 to 60 minutes of operation.	For the 3C920V1-005-based systems, it has been discovered that the retry of a PCI master cycle gets terminated with the assertion of SERRN because the intended operation exceeds the PCI time-out interval of 1ms. Some PCI chipsets, when enabled, will generate an NMI (non-maskable interrupt) associated with this time-out. If the agent does not want an NMI to be generated, a different reporting mechanism is employed. This could also be translated into either a performance issue or an inequitable Tx/Rx load-sharing problem. There are two possible causes of the time-out. One is due to the retried PCI master cycle gets locked out by the 3C920V1's internal Tx/Rx arbitration. The other is caused by the race condition that exists between generating a bus request before the FIFO free space has been updated. The Tx/Rx arbitration did not take into account the possibility of retried PCI master cycles or other network conditions causing Tx and Rx to lock each other out. This eventually results in a system time-out. The race condition exists when a PCI bus request is issued prematurely prior to the FIFO's space being updated. This becomes a problem when the space in the FIFO reaches a burst threshold boundary. Under these circumstances, a bus request is not re-generated. As the space in the FIFO is freed up, the PCI bus request is re-issued. However, if the retried PCI bus cycle does not complete before the system time-out requirement, the assertion of SERRN will occur as a consequenceof the time-out.	There is currently no software or hardware workaround available. The 3C920V1-006 revision is presently in process to eliminate this problem.eliminate this problem.

(2 of 2)

INDEX

INDEX OF REGISTERS

T

U

V

INDEX OF BITS

D

E

F

G

H

U

V

W

X