PRELIMINARY

Am79C970

PCnet™-PCI Single-Chip Ethernet Controller for PCI Local Bus

Advanced Micro Devices

DISTINCTIVE CHARACTERISTICS

- **Single-chip Ethernet controller for the Peripheral Component Interconnect (PCI) local bus**
- **Supports ISO 8802-3 (IEEE/ANSI 802.3) and Ethernet Standards**
- **Direct interface to the PCI local bus**
- **Compliant to PCI local bus specification (Revision 2.0)**
- **Software compatible with AMD's Am7990 LANCE, Am79C90 C-LANCE, Am79C960 PCnet-ISA, Am79C961 PCnet-ISA+, Am79C965 PCnet-32, and Am79C900 ILACC™ register and descriptor architecture**
- **Compatible with Am2100/Am1500T and Novell® NE2100/NE1500 driver software**
- **High-performance Bus Master architecture with integrated DMA Buffer Management Unit for low CPU and bus utilization**
- **Big endian byte alignment supported**
- **Single +5 V power supply operation**
- **Low-power, CMOS design with sleep modes allows reduced power consumption for critical battery powered applications and Green PCs**
- **Microwire™ EEPROM interface supports jumperless design**
- **Individual 136-byte transmit and 128-byte receive FIFOs provide frame buffering for increased system latency, and support the following features:**
  - — Automatic retransmission with no FIFO reload
  - — Automatic receive stripping and transmit padding (individually programmable)
  - — Automatic runt packet rejection
  - — Automatic deletion of received collision frames
- **Look-Ahead Packet Processing (LAPP) concept allows protocol analysis to begin before end of receive frame**
- **Integrated Manchester Encoder/Decoder**
- **Provides integrated Attachment Unit Interface (AUI) and 10BASE-T transceiver with automatic port selection**
- **Automatic Twisted-Pair receive polarity detection and automatic correction of the receive polarity**
- **Optional byte padding to long-word boundary on receive**
- **Dynamic transmit FCS generation programmable on a frame-by-frame basis**
- **Internal/external loopback capabilities**
- **Supports the following types of network interfaces:**
  - — AUI to external 10BASE2, 10BASE5,10BASE-T or 10BASE-F MAU
  - — Internal 10BASE-T transceiver with Smart Squelch to Twisted-Pair medium
- **NAND Tree test mode for connectivity testing on printed circuit boards**
- **132-pin PQFP package**

GENERAL DESCRIPTION

The PCnet-PCI single-chip 32-bit Ethernet controller is a highly integrated Ethernet system solution designed to address high-performance system application requirements. It is a flexible bus-mastering device that can be used in any application, including network-ready PCs, printers, fax modems, and bridge/router designs. The bus-master architecture provides high data throughput in the system and low CPU and system bus utilization. The PCnet-PCI controller is fabricated with AMD's advanced low-power CMOS process to provide low operating and standby current for power sensitive applications.

The PCnet-PCI controller is a complete Ethernet node integrated into a single VLSI device. It contains a bus interface unit, a DMA buffer management unit, an IEEE 802.3-defined Media Access Control (MAC) function, individual 136-byte transmit and 128-byte receive FIFOs, an IEEE 802.3-defined Attachment Unit Interface (AUI) and Twisted-Pair Transceiver Media Attachment Unit (10BASE-T MAU), and a Microwire EEPROM interface. The PCnet-PCI controller is also register compatible with the LANCE (Am7990) Ethernet controller, the C-LANCE (Am79C90) Ethernet controller, the ILACC (Am79C900) Ethernet controller, and all Ethernet

This document contains information on a product under development at Advanced Micro Devices, Inc. The information is intended to help you to evaluate this product. AMD reserves the right to change or discontinue work on this proposed product without notice.

Publication# **18220** Rev. **C** Amendment **/0**
Issue Date: **June 1994**

controllers in the PCnet Family, including the PCnet-ISA controller (Am79C960), the PCnet-ISA+ controller (Am79C961), and the PCnet-32 controller (Am79C965). The buffer management unit supports the LANCE, ILACC, and PCnet descriptor software models. The PCnet-PCI controller is software compatible with the Novell NE2100 and NE1500 Ethernet adapter card architectures. In addition, a Sleep function has been incorporated to provide low standby current, excellent for notebooks and Green PCs.

The 32-bit multiplexed bus interface unit provides a direct interface to the PCI local bus applications, simplifying the design of an Ethernet node in a PC system. With its built-in support for both little and big endian byte alignment, this controller also addresses proprietary non-PC applications.

The PCnet-PCI controller supports auto configuration in the PCI configuration space. Additional PCnet-PCI controller configuration parameters, including the unique IEEE physical address, can be read from an external non-volatile memory (serial EEPROM) immediately following system RESET.

The controller also has the capability to automatically select either the AUI port or the Twisted-Pair transceiver. Only one interface is active at any one time. The individual transmit and receive FIFOs optimize system overhead, providing sufficient latency during frame transmission and reception, and minimizing intervention during normal network error recovery. The integrated Manchester encoder/decoder (MENDEC) eliminates the need for an external Serial Interface Adapter (SIA) in the system. In addition, the device provides programmable on-chip LED drivers for transmit, receive, collision, receive polarity, link integrity or jabber status.

BLOCK DIAGRAM

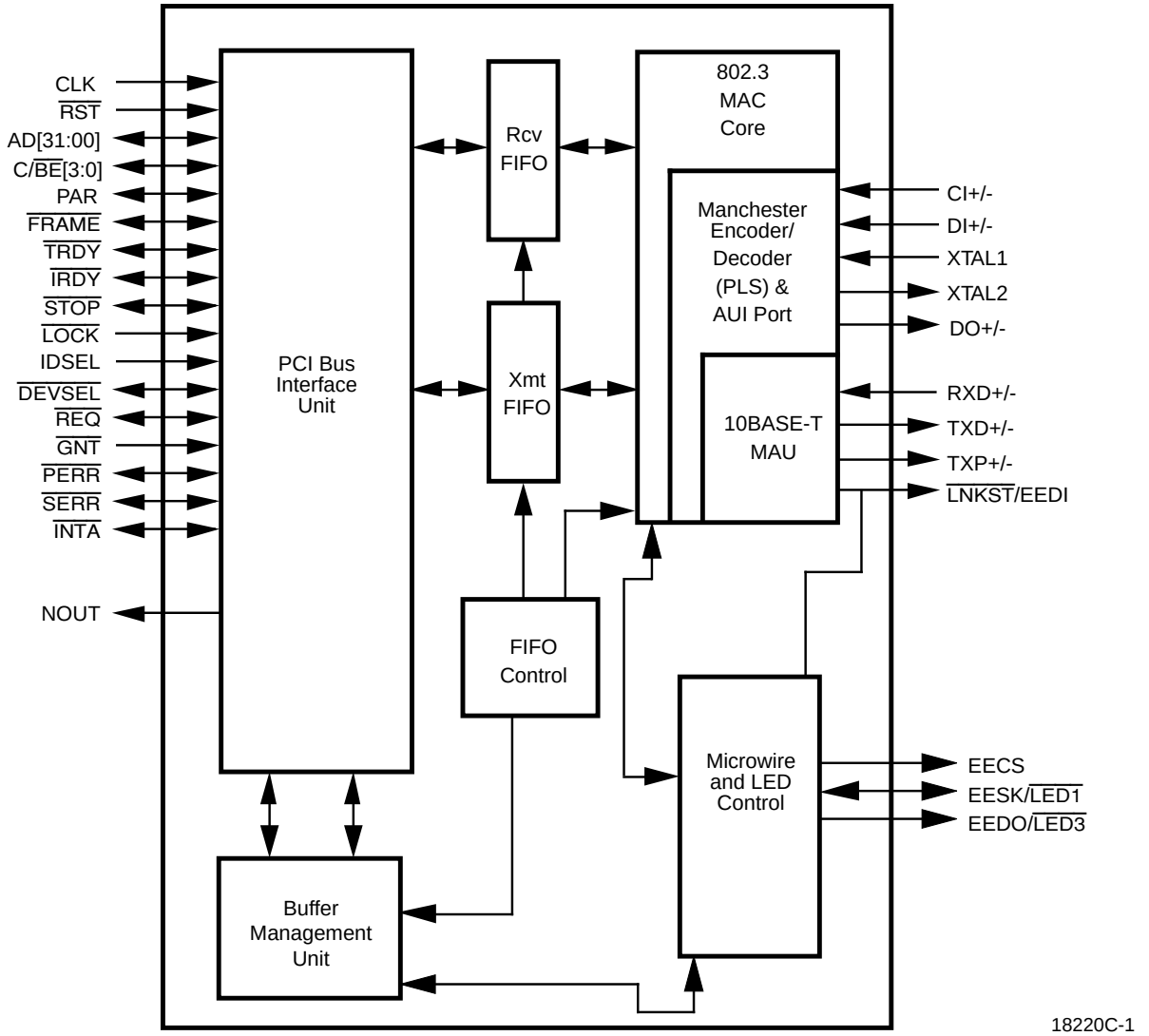


TABLE OF CONTENTS

DISTINCTIVE CHARACTERISTICS 1-868
GENERAL DESCRIPTION 1-868
BLOCK DIAGRAM 1-869
RELATED PRODUCTS 1-870
CONNECTION DIAGRAM 1-871
ORDERING INFORMATION 1-872
PIN DESIGNATIONS 1-879
Listed By Pin Number 1-879
Listed By Group 1-880
Driver Type 1-881
PIN DESCRIPTION 1-882
PCI Interface 1-882
Board Interface 1-884
Microwire EEPROM Interface 1-885
Attachment Unit Interface 1-886
Twisted-Pair Interface 1-886
Test Interface 1-886
Power Supply Pins 1-886
BASIC FUNCTIONS 1-887
System Bus Interface Function 1-887
Software Interface 1-887
Network Interfaces 1-887
DETAILED FUNCTIONS 1-888
Bus Interface Unit (BIU) 1-888
Bus Acquisition 1-888
Bus Master DMA Transfers 1-889
Target Initiated Termination 1-894
Master Initiated Termination 1-897
Initialization Block DMA Transfers 1-900
Descriptor DMA Transfers 1-901
FIFO DMA Transfers 1-904
Slave I/O Transfers 1-909
Slave Configuration Transfers 1-911
Buffer Management Unit (BMU) 1-913
Initialization 1-913
Re-Initialization 1-913
Buffer Management 1-913
Descriptor Rings 1-913
Descriptor Ring Access Mechanism 1-914
Polling 1-916
Transmit Descriptor Table Entry (TDTE) 1-916
Receive Descriptor Table Entry (RDTE) 1-918
Media Access Control 1-918
Transmit and Receive Message Data Encapsulation 1-918
Media Access Management 1-920
Manchester Encoder/Decoder (MENDEC) 1-922
External Crystal Characteristics 1-922
External Clock Drive Characteristics 1-922

MENDEC Transmit Path 1-922
Transmitter Timing and Operation 1-922
Receiver Path 1-922
Input Signal Conditioning 1-922
Clock Acquisition 1-922
PLL Tracking 1-922
Carrier Tracking and End of Message 1-924
Data Decoding 1-924
Differential Input Terminations 1-924
Collision Detection 1-925
Jitter Tolerance Definition 1-925
Attachment Unit Interface (AUI) 1-925

Twisted-Pair Transceiver (T-MAU) 1-925

Twisted-Pair Transmit Function 1-925
Twisted-Pair Receive Function 1-925
Link Test Function 1-925
Polarity Detection and Reversal 1-926
Twisted-Pair Interface Status 1-926
Collision Detect Function 1-927
Signal Quality Error (SQE) Test (Heartbeat) Function 1-927
Jabber Function 1-927
Power Down 1-927
10BASE-T Interface Connection 1-927

Power Savings Modes 1-928

Software Access 1-928

Configuration Registers 1-928
I/O Resources 1-929
I/O Register Access 1-931

Hardware Access 1-933

PCnet-PCI Controller Master Accesses 1-933
Slave Access to I/O Resources 1-934
EEPROM Microwire Access 1-935

Transmit Operation 1-938

Transmit Function Programming 1-938
Automatic Pad Generation 1-938
Transmit FCS Generation 1-939
Transmit Exception Conditions 1-939

Receive Operation 1-940

Receive Function Programming 1-940
Automatic Pad Stripping 1-940
Receive FCS Checking 1-941
Receive Exception Conditions 1-941

Loopback Operation 1-941

LED Support 1-942

H_RESET, S_RESET, and STOP 1-943

H_RESET 1-943
S_RESET 1-943
STOP 1-943

NAND Tree Testing 1-944

USER ACCESSIBLE REGISTERS 1-947
PCI Configuration Registers 1-948
Vendor ID 1-948
Device ID Register 1-948
Command Register 1-949
Status Register 1-949
Revision ID Register 1-951
Programming Interface Register 1-951
Sub-Class Register 1-951
Base-Class Register 1-951
Latency Timer Register 1-951
Header Type Register 1-951
Base Address Register 1-951
Interrupt Line Register 1-952
Interrupt Pin Register 1-952
RAP Register 1-952
RAP: Register Address Port 1-952
Control and Status Registers 1-952
CSR0: PCnet-PCI Controller Status Register 1-952
CSR1: IADR[15:0] 1-955
CSR2: IADR[31:16] 1-955
CSR3: Interrupt Masks and Deferral Control 1-955
CSR4: Test and Features Control 1-957
CSR6: RX/TX Descriptor Table Length 1-959
CSR8: Logical Address Filter, LADRF[15:0] 1-960
CSR9: Logical Address Filter, LADRF[31:16] 1-960
CSR10: Logical Address Filter, LADRF[47:32] 1-960
CSR11: Logical Address Filter, LADRF[63:48] 1-960
CSR12: Physical Address Register, PADR[15:0] 1-960
CSR13: Physical Address Register, PADR[31:16] 1-960
CSR14: Physical Address Register, PADR[47:32] 1-961
CSR15: Mode Register 1-961
CSR16: Initialization Block Address Lower 1-963
CSR17: Initialization Block Address Upper 1-963
CSR18: Current Receive Buffer Address Lower 1-963
CSR19: Current Receive Buffer Address Upper 1-963
CSR20: Current Transmit Buffer Address Lower 1-963
CSR21: Current Transmit Buffer Address Upper 1-963
CSR22: Next Receive Buffer Address Lower 1-963
CSR23: Next Receive Buffer Address Upper 1-963
CSR24: Base Address of Receive Ring Lower 1-964
CSR25: Base Address of Receive Ring Upper 1-964
CSR26: Next Receive Descriptor Address Lower 1-964
CSR27: Next Receive Descriptor Address Upper 1-964
CSR28: Current Receive Descriptor Address Lower 1-964
CSR29: Current Receive Descriptor Address Upper 1-964
CSR30: Base Address of Transmit Ring Lower 1-964
CSR31: Base Address of Transmit Ring Upper 1-964
CSR32: Next Transmit Descriptor Address Lower 1-965
CSR33: Next Transmit Descriptor Address Upper 1-965
CSR34: Current Transmit Descriptor Address Lower 1-965
CSR35: Current Transmit Descriptor Address Upper 1-965
CSR36: Next Receive Descriptor Address Lower 1-965
CSR37: Next Receive Descriptor Address Upper 1-965

CSR38: Next Transmit Descriptor Address Lower 1-965
CSR39: Next Transmit Descriptor Address Upper 1-965
CSR40: Current Receive Status and Byte Count Lower 1-966
CSR41: Current Receive Status and Byte Count Upper 1-966
CSR42: Current Transmit Status and Byte Count Lower 1-966
CSR43: Current Transmit Status and Byte Count Upper 1-966
CSR44: Next Receive Status and Byte Count Lower 1-966
CSR45: Next Receive Status and Byte Count Upper 1-966
CSR46: Poll Time Counter 1-967
CSR47: Polling Interval 1-967
CSR58: Software Style 1-967
CSR59: IR Register 1-968
CSR60: Previous Transmit Descriptor Address Lower 1-969
CSR61: Previous Transmit Descriptor Address Upper 1-969
CSR62: Previous Transmit Status and Byte Count Lower 1-969
CSR63: Previous Transmit Status and Byte Count Upper 1-969
CSR64: Next Transmit Buffer Address Lower 1-969
CSR65: Next Transmit Buffer Address Upper 1-969
CSR66: Next Transmit Status and Byte Count Lower 1-969
CSR67: Next Transmit Status and Byte Count Upper 1-970
CSR72: Receive Ring Counter 1-970
CSR74: Transmit Ring Counter 1-970
CSR76: Receive Ring Length 1-970
CSR78: Transmit Ring Length 1-970
CSR80: DMA Transfer Counter and FIFO Threshold Control 1-970
CSR82: Bus Activity Timer 1-972
CSR84: DMA Address Register Lower 1-973
CSR85: DMA Address Register Upper 1-973
CSR86: Buffer Byte Counter 1-973
CSR88: Chip ID Register Lower 1-973
CSR89: Chip ID Register Upper 1-974
CSR92: Ring Length Conversion 1-974
CSR94: Transmit Time Domain Reflectometry Count 1-974
CSR100: Bus Timeout 1-974
CSR112: Missed Frame Count 1-974
CSR114: Receive Collision Count 1-975
CSR122: Receive Frame Alignment Control 1-975
CSR124: Buffer Management Test (BMU) Register 1-975

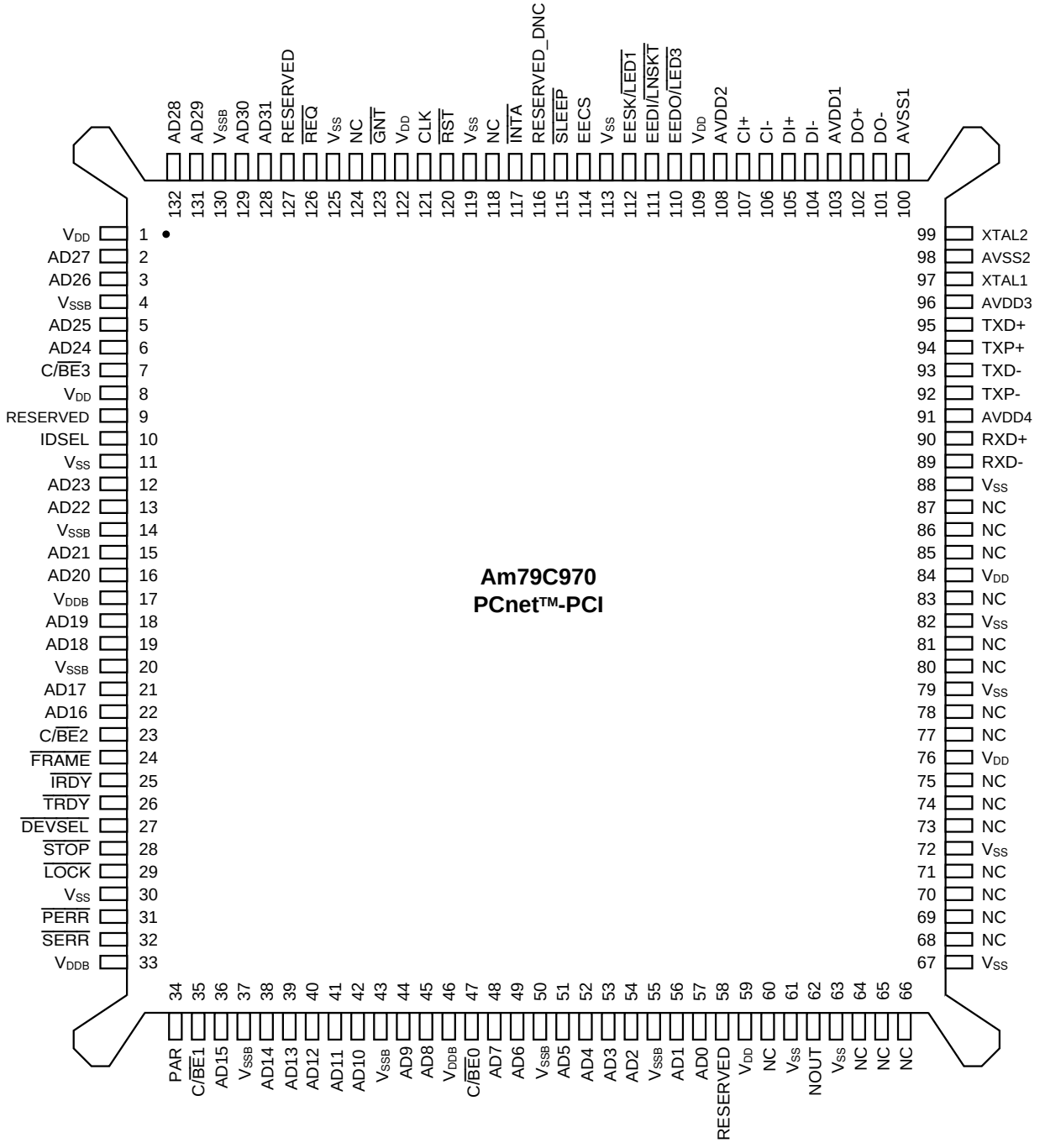
Bus Configuration Registers 1-976
BCR0: Master Mode Read Active 1-977
BCR1: Master Mode Write Active 1-977
BCR2: Miscellaneous Configuration 1-977
BCR4: Link Status LED (LNKST) 1-978
BCR5: LED1 Status 1-979
BCR6: LED2 Status 1-980
BCR7: LED3 Status 1-980
BCR16: I/O Base Address Lower 1-981
BCR17: I/O Base Address Upper 1-982
BCR18: Burst Size and Bus Control Register 1-982
BCR19: EEPROM Control and Status Register 1-984
BCR20: Software Style 1-987
BCR21: Interrupt Control 1-988
Initialization Block 1-989
RLEN and TLEN 1-989
RDRA and TDRA 1-990
LADRF 1-990
PADR 1-991
MODE 1-991
Receive Descriptors 1-991
RMD0 1-992
RMD1 1-992
RMD2 1-993
RMD3 1-993
Transmit Descriptors 1-994
TMD0 1-994
TMD1 1-994
TMD2 1-995
TMD3 1-996
Register Summary 1-997
Control and Status Registers 1-997
BCR — Bus Configuration Registers 1-1000

ABSOLUTE MAXIMUM RATINGS 1-1001
OPERATING RANGES 1-1001
DC CHARACTERISTICS 1-1001
SWITCHING CHARACTERISTICS: **Bus Interface** 1-1004
SWITCHING CHARACTERISTICS: **10BASE-T Interface** 1-1005
SWITCHING CHARACTERISTICS: **Attachment Unit Interface** 1-1006
KEY TO SWITCHING WAVEFORMS 1-1007
SWITCHING TEST CIRCUITS 1-1008
SWITCHING WAVEFORMS: **System Bus Interface** 1-1009
SWITCHING WAVEFORMS: **10BASE-T Interface** 1-1011
SWITCHING WAVEFORMS: **Attachment Unit Interface** 1-1013
APPENDIX A: PCnet-PCI Compatible Media Interface Modules 1-1016
APPENDIX B: Recommendation for Power and Ground Decoupling 1-1019
APPENDIX C: Alternative Method for Initialization 1-1021
APPENDIX D: Look-Ahead Packet Processing (LAPP) Concept 1-1022
DATA SHEET REVISION SUMMARY 1-1032

RELATED PRODUCTS

Part No.	Description
Am79C98	Twisted-Pair Ethernet Transceiver (TPEX)
Am79C100	Twisted-Pair Ethernet Transceiver Plus (TPEX+)
Am7996	IEEE 802.3/Ethernet/Cheapernet Tap Transceiver
Am79C981	Integrated Multiport Repeater Plus™ (IMR+™)
Am79C987	Hardware Implemented Management Information Base™ (HIMIB™)
Am79C940	Media Access Controller for Ethernet (MACE™)
Am7990	Local Area Network Controller for Ethernet (LANCE)
Am79C90	CMOS Local Area Network Controller for Ethernet (C-LANCE)
Am79C900	Integrated Local Area Communications Controller™ (ILACC™)
Am79C960	PCnet-ISA Single-Chip Ethernet Controller (for ISA bus)
Am79C961	PCnet-ISA+ Single-Chip Ethernet Controller (with Microsoft® Plug n' Play support)
Am79C965	PCnet-32 Single-Chip 32-Bit Ethernet Controller (for 486 and VL buses)
Am79C974	PCnet-SCSI Combination Ethernet and SCSI Controller for PCI Systems

CONNECTION DIAGRAM



18220C-2

Pin 1 is marked for orientation.

NC = No Connection, reserved for future use.

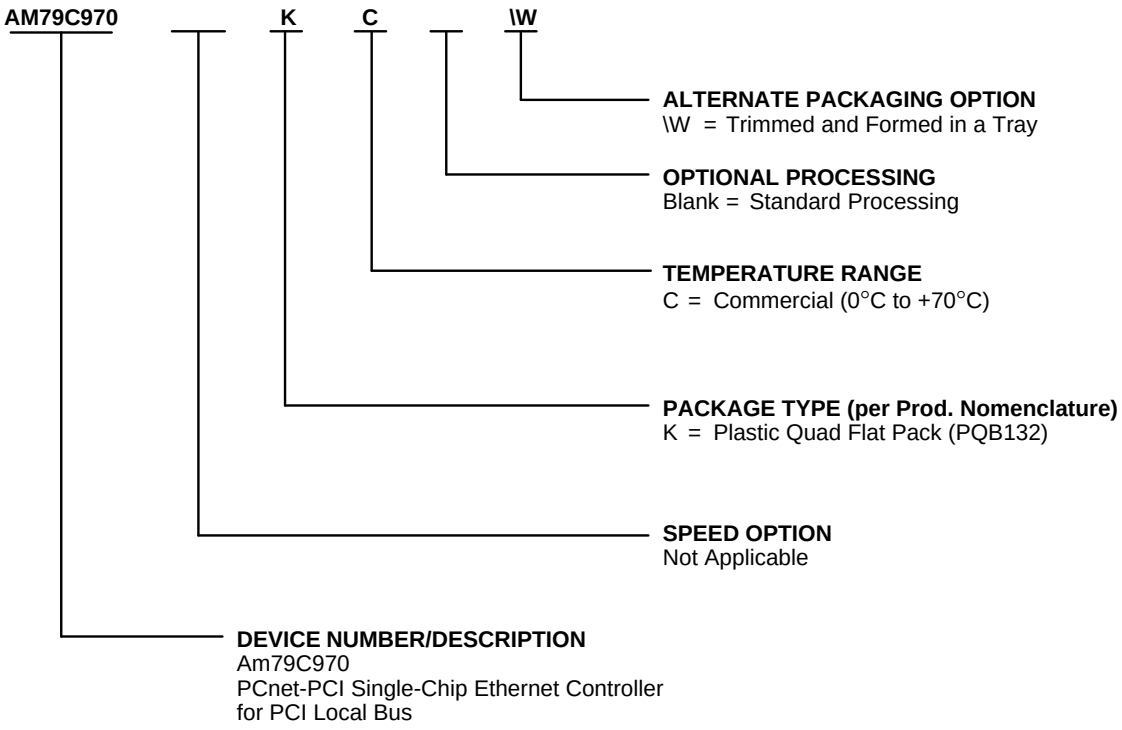
RESERVED = Internally bonded, for AMD internal test only; should not be connected.

RESERVED_DNC = Reserved, don't connect.

ORDERING INFORMATION

Standard Products

AMD standard products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of:



Valid Combinations	
AM79C970	KC, KC\W

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations and to check on newly released combinations.

PIN DESIGNATIONS

Listed by Pin Number

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
1	V_{DDB}	34	PAR	67	V_{SS}	100	AVSS1
2	AD27	35	C/$\overline{BE}$1	68	NC	101	DO–
3	AD26	36	AD15	69	NC	102	DO+
4	V_{SSB}	37	V_{SSB}	70	NC	103	AVDD1
5	AD25	38	AD14	71	NC	104	DI–
6	AD24	39	AD13	72	V_{SS}	105	DI+
7	C/$\overline{BE}$3	40	AD12	73	NC	106	CI–
8	V_{DD}	41	AD11	74	NC	107	CI+
9	RESERVED	42	AD10	75	NC	108	AVDD2
10	IDSEL	43	V_{SSB}	76	V_{DD}	109	V_{DD}
11	V_{SS}	44	AD9	77	NC	110	EEDO/$\overline{LED3}$
12	AD23	45	AD8	78	NC	111	EEDI/$\overline{LNKST}$
13	AD22	46	V_{DDB}	79	V_{SS}	112	EESK/$\overline{LED1}$
14	V_{SSB}	47	C/$\overline{BE}$0	80	NC	113	V_{SS}
15	AD21	48	AD7	81	NC	114	EECS
16	AD20	49	AD6	82	V_{SS}	115	$\overline{SLEEP}$
17	V_{DDB}	50	V_{SSB}	83	NC	116	RESERVED_DNC
18	AD19	51	AD5	84	V_{DD}	117	$\overline{INTA}$
19	AD18	52	AD4	85	NC	118	NC
20	V_{SSB}	53	AD3	86	NC	119	V_{SS}
21	AD17	54	AD2	87	NC	120	$\overline{RST}$
22	AD16	55	V_{SSB}	88	V_{SS}	121	CLK
23	C/$\overline{BE}$2	56	AD1	89	RXD–	122	V_{DD}
24	$\overline{FRAME}$	57	AD0	90	RXD+	123	$\overline{GNT}$
25	$\overline{IRDY}$	58	RESERVED	91	AVDD4	124	RESERVED
26	$\overline{TRDY}$	59	V_{DD}	92	TXP–	125	V_{SS}
27	$\overline{DEVSEL}$	60	NC	93	TXD–	126	$\overline{REQ}$
28	$\overline{STOP}$	61	V_{SS}	94	TXP+	127	RESERVED
29	$\overline{LOCK}$	62	NOUT	95	TXD+	128	AD31
30	V_{SS}	63	V_{SS}	96	AVDD3	129	AD30
31	$\overline{PERR}$	64	NC	97	XTAL1	130	V_{SSB}
32	$\overline{SERR}$	65	NC	98	AVSS2	131	AD29
33	V_{DDB}	66	NC	99	XTAL2	132	AD28

PIN DESIGNATIONS

Listed by Group

Pin Name	Pin Function	Type	Driver	# Pins
PCI Bus Interface				
AD[31:00]	Address/Data Bus	IO	TS3	32
C/$\overline{BE}$[3:0]	Bus Command/Byte Enable	IO	TS3	4
CLK	Bus Clock	I	NA	1
$\overline{DEVSEL}$	Device Select	IO	TS6	1
$\overline{FRAME}$	Cycle Frame	IO	TS6	1
$\overline{GNT}$	Bus Grant	I	NA	1
IDSEL	Initialization Device Select	I	ns	1
$\overline{INTA}$	Interrupt	IO	OD6	1
$\overline{IRDY}$	Initiator Ready	IO	TS6	1
$\overline{LOCK}$	Bus Lock	I	NA	1
PAR	Parity	IO	TS6	1
$\overline{PERR}$	Parity Error	IO	TS6	1
$\overline{REQ}$	Bus Request	IO	TS3	1
$\overline{RST}$	Reset	I	NA	1
$\overline{SERR}$	System Error	IO	OD6	1
$\overline{STOP}$	Stop	IO	TS6	1
$\overline{TRDY}$	Target Ready	IO	TS6	1
Board Interface				
EECS	Microwire Serial PROM Chip Select	O	O8	1
EEDI/$\overline{LNKST}$	Microwire Serial EEPROM Data In/Link Status	O	LED	1
EEDO/$\overline{LED3}$	Microwire APROM Data Out/LED predriver	IO	LED	1
EESK/$\overline{LED1}$	Microwire Serial PROM Clock/LED1	IO	LED	1
SLEEP	Sleep Mode	I	NA	1
XTAL1–2	Crystal Input/Output	IO	NA	2
Attachment Unit Interface (AUI)				
CI+/CI–	AUI Collision Differential Pair	I	NA	2
DI+/DI–	AUI Data In Differential Pair	I	NA	2
DO+/DO–	AUI Data Out Differential Pair	O	DO	2
10BASE-T Interface				
RXD+/RXD–	Receive Differential Pair	I	NA	2
TXD+/TXD–	Transmit Differential Pair	O	TDO	2
TXP+/TXP–	Transmit Pre-distortion Differential Pair	O	TPO	2
$\overline{LNKST}$/EEDI	Link Status/Microwire Serial EEPROM Data In	O	LED	1
Test Interface				
NOUT	NAND Tree Test Output	O	O3	1
Power Supplies				
AVDD	Analog Power	P	NA	4
AVSS	Analog Ground	P	NA	2
V_{DD}	Digital Power	P	NA	6
V_{SS}	Digital Ground	P	NA	12
V_{DDB}	I/O Buffer Power	P	NA	4
V_{SSB}	I/O Buffer Ground	P	NA	8

PIN DESIGNATIONS

Listed by Driver Type

The next table describes the various types of drivers that are implemented in the PCnet-PCI controller. Current is given as milliamperes:

Name	Type	I_{OL} (mA)	I_{OH} (mA)	pF
TS3	Tri-State™	3	–2	50
TS6	Tri-State	6	–2	50
O3	Totem Pole	3	–0.4	50
O8	Totem Pole	8	–0.4	50
OD6	Open Drain	6	NA	50
LED	LED	12	–0.4	50

PIN DESCRIPTION

PCI Interface

AD[31:00]

Address and Data Input/Output

These signals are multiplexed on the same PCI pins. During the first clock of a transaction AD[31:00] contain the physical byte address (32 bits). During the subsequent clocks AD[31:00] contain data. Byte ordering is little endian by default. AD[07:00] are defined as least significant byte and AD[31:24] are defined as the most significant byte. For FIFO data transfers, the PCnet-PCI controller can be programmed for big endian byte ordering. See CSR3, bit 2 (BSWP) for more details.

During the address phase of the transaction, when the PCnet-PCI controller is a bus master, AD[31:2] will address the active DWORD (double-word). The PCnet-PCI controller always drives AD[1:0] to '00' during the address phase indicating linear burst order. When the PCnet-PCI controller is not a bus master, the AD[31:00] lines are continuously monitored to determine if an address match exists for I/O slave transfers.

During the data phase of the transaction, AD[31:00] are driven by the PCnet-PCI controller when performing bus master writes and slave read operations. Data on AD[31:00] is latched by the PCnet-PCI controller when performing bus master reads and slave write operations.

When $\overline{RST}$ is active, AD[31:0] are inputs for NAND tree testing.

C/$\overline{BE}$[3:0]

Bus Command and Byte Enables
Input/Output

These signals are multiplexed on the same PCI pins. During the address phase of the transaction, C/$\overline{BE}$[3:0] define the bus command. During the data phase C/$\overline{BE}$[3:0] are used as Byte Enables. The Byte Enables define which physical byte lanes carry meaningful data. C/$\overline{BE}$0 applies to byte 0 (AD[07:00]) and C/$\overline{BE}$3 applies to byte 3 (AD[31:24]). The function of the Byte Enables is independent of the byte ordering mode (CSR3, bit 2).

When $\overline{RST}$ is active, C/$\overline{BE}$[3:0] are inputs for NAND tree testing.

CLK

Clock
Input

This signal provides timing for all the transactions on the PCI bus and all PCI devices on the bus including the PCnet-PCI controller. All bus signals are sampled on the rising edge of CLK and all parameters are defined with respect to this edge. The PCnet-PCI controller operates over a range of 0 to 33 MHz.

When $\overline{RST}$ is active, CLK is an input for NAND tree testing.

$\overline{DEVSEL}$

Device Select
Input/Output

This signal when actively driven by the PCnet-PCI controller as a slave device signals to the master device that the PCnet-PCI controller has decoded its address as the target of the current access. As an input it indicates whether any device on the bus has been selected.

When $\overline{RST}$ is active, $\overline{DEVSEL}$ is an input for NAND tree testing.

$\overline{FRAME}$

Cycle Frame
Input/Output

This signal is driven by the PCnet-PCI controller when it is the bus master to indicate the beginning and duration of the access. $\overline{FRAME}$ is asserted to indicate a bus transaction is beginning. $\overline{FRAME}$ is asserted while data transfers continue. $\overline{FRAME}$ is deasserted when the transaction is in the final data phase.

When $\overline{RST}$ is active, $\overline{FRAME}$ is an input for NAND tree testing.

$\overline{GNT}$

Bus Grant
Input

This signal indicates that the access to the bus has been granted to the PCnet-PCI controller.

The PCnet-PCI controller supports bus parking. When the PCI bus is idle and the system arbiter asserts $\overline{GNT}$ without an active $\overline{REQ}$ from the PCnet-PCI controller, the PCnet-PCI controller will actively drive the AD, C/$\overline{BE}$ and PAR lines.

When $\overline{RST}$ is active, $\overline{GNT}$ is an input for NAND tree testing.

IDSEL

Initialization Device Select
Input

This signal is used as a chip select for the PCnet-PCI controller during configuration read and write transaction.

When $\overline{RST}$ is active, IDSEL is an input for NAND tree testing.

INTA
Interrupt Request
Input/Output

An asynchronous attention signal which indicates that one or more of the following status flags is set: BABL, MISS, MERR, RINT, IDON, RCVCCO, RPCO, JAB, MPCO, or TXSTRT. Each status flag has a mask bit which allows for suppression of INTA assertion. The flags have the following meaning:

BABL	Babble
RCVCCO	Receive Collision Count Overflow
RPCO	Runt Packet Count Overflow
JAB	Jabber
MISS	Missed Frame
MERR	Memory Error
MPCO	Missed Packet Count Overflow
RINT	Receive Interrupt
IDON	Initialization Done
TXSTRT	Transmit Start

When RST is active, INTA is an input for NAND tree testing.

IRDY
Initiator Ready
Input/Output

This signal indicates PCnet-PCI controllers ability, as a master device, to complete the current data phase of the transaction. IRDY is used in conjunction with the TRDY. A data phase is completed on any clock when both IRDY and TRDY are asserted. During a write IRDY indicates that valid data is present on AD[31:00]. During a read IRDY indicates that data is accepted by the PCnet-PCI controller as a bus master. Wait states are inserted until both IRDY and TRDY are asserted simultaneously.

When RST is active, IRDY is an input for NAND tree testing.

LOCK
Lock
Input

LOCK is used by the current bus master to indicate an atomic operation that may require multiple transfers.

As a slave device, the PCnet-PCI controller can be locked by any master device. When another master attempts to access the PCnet-PCI while it is locked, the PCnet-PCI controller will respond by asserting DEVSEL and STOP with TRDY deasserted (PCI retry).

The PCnet-PCI controller will never assert LOCK as a master.

When RST is active, LOCK is an input for NAND tree testing.

PAR
Parity
Input/Output

Parity is even parity across AD[31:00] and C/BE[3:0].When the PCnet-PCI controller is a bus master, it generates parity during the address and write data phases. It checks parity during read data phases. When the PCnet-PCI controller operates in slave mode and is the target of the current cycle, it generates parity during read data phases. It checks parity during address and write data phases.

When RST is active, PAR is an input for NAND tree testing.

PERR
Parity Error
Input/Output

This signal is asserted by the PCnet-PCI controller when it checks for parity error during any data phase when its AD[31:00] lines are inputs. The PERR pin is only active when PERREN (bit 6) in the PCI command register is set.

The PCnet-PCI controller monitors the PERR input during a bus master write cycle. It will assert the Data Parity Reported bit in the Status register of the Configuration Space when a parity error is reported by the target device.

When RST is active, PERR is an input for NAND tree testing.

REQ
Bus Request
Input/Output

The PCnet-PCI controller asserts REQ pin as a signal that it wishes to become a bus master. Once asserted, REQ remains active until GNT has become active, independent of subsequent assertion of SLEEP or setting of the STOP bit or access to the S_RESET port (offset 14h).

When RST is active, REQ is an input for NAND tree testing.

RST
Reset
Input

When RST is asserted low, then the PCnet-PCI controller performs an internal system reset of the type H_RESET (HARDWARE_RESET). RST must be held for a minimum of 30 CLK periods. While in the H_RESET state, the PCnet-PCI controller will disable or deassert all outputs. RST may be asynchronous to the CLK when asserted or deasserted. It is recommended that the deassertion be synchronous to guarantee clean and bounce free edge.

When $\overline{RST}$ is active, NAND tree testing is enabled. All PCI interface pins are in input mode. The result of the NAND tree testing can be observed on the NOUT output (pin 62).

$\overline{SERR}$
System Error
Input/Output

This signal is asserted for one CLK by the PCnet-PCI controller when it detects a parity error during the address phase when its AD[31:00] lines are inputs.

The $\overline{SERR}$ pin is only active when SERREN (bit 8) and PERREN (bit 6) in the PCI command register are set.

When $\overline{RST}$ is active, $\overline{SERR}$ is an input for NAND tree testing.

$\overline{STOP}$
Stop
Input/Output

In the slave role, the PCnet-PCI controller drives the $\overline{STOP}$ signal to inform the bus master to stop the current transaction. In the bus master role, the PCnet-PCI controller receives the $\overline{STOP}$ signal and stops the current transaction.

When $\overline{RST}$ is active, $\overline{STOP}$ is an input for NAND tree testing.

$\overline{TRDY}$
Target Ready
Input/Output

This signal indicates that the PCnet-PCI controllers ability as a selected device to complete the current data phase of the transaction. $\overline{TRDY}$ is used in conjunction with the $\overline{IRDY}$. A data phase is completed on any clock both $\overline{TRDY}$ and $\overline{IRDY}$ are asserted. During a read $\overline{TRDY}$ indicates that valid data is present on AD[31:00]. During a write, $\overline{TRDY}$ indicates that data has been accepted. Wait states are inserted until both $\overline{IRDY}$ and $\overline{TRDY}$ are asserted simultaneously.

When $\overline{RST}$ is active, $\overline{TRDY}$ is an input for NAND tree testing.

Board Interface

$\overline{LED1}$
LED1
Output

This pin is shared with the EESK function. As $\overline{LED1}$, the function and polarity of this pin are programmable through BCR5. By default, $\overline{LED1}$ is active LOW and it indicates receive activity on the network. The $\overline{LED1}$ output from the PCnet-PCI controller is capable of sinking the necessary 12 mA of current to drive an LED directly.

The $\overline{LED1}$ pin is also used during EEPROM Auto-detection to determine whether or not an EEPROM is present at the PCnet-PCI controller Microwire interface. At the trailing edge of the $\overline{RST}$ pin, $\overline{LED1}$ is sampled to determine the value of the EEDET bit in BCR19. A sampled HIGH value means that an EEPROM is present, and EEDET will be set to ONE. A sampled LOW value means that an EEPROM is not present, and EEDET will be set to ZERO. See the EEPROM Auto-detection section for more details.

If no LED circuit is to be attached to this pin, then a pull up or pull down resistor must be attached instead, in order to resolve the EEDET setting.

$\overline{LED3}$
LED3
Output

This pin is shared with the EEDO function. When functioning as $\overline{LED3}$, the signal on this pin is programmable through BCR7. By default, $\overline{LED3}$ is active LOW and it indicates transmit activity on the network. Special attention must be given to the external circuitry attached to this pin. If an LED circuit were directly attached to this pin, it would create an I_{OL} requirement that could not be met by the serial EEPROM that would also be attached to this pin. (This pin is multifunctioned with the EEDO function of the Microwire serial EEPROM interface.)

Therefore, if this pin is to be used as an additional LED output while an EEPROM is used in the system, then buffering is required between the $\overline{LED3}$ pin and the LED circuit. If no EEPROM is included in the system design, then the $\overline{LED3}$ signal may be directly connected to an LED without buffering. The $\overline{LED3}$ output from the PCnet-PCI controller is capable of sinking the necessary 12 mA of current to drive an LED in this case. For more details regarding LED connection, see the section on LEDs.

$\overline{LNKST}$
LINK Status
Output

This pin provides 12 mA for driving an LED. By default, it indicates an active link connection on the 10BASE-T interface. This pin can also be programmed to indicate other network status (see BCR4). The $\overline{LNKST}$ pin polarity is programmable, but by default, it is active LOW. Note that this pin is multiplexed with the EEDI function.

$\overline{SLEEP}$
Sleep
Input

When $\overline{SLEEP}$ is asserted (active LOW), the PCnet-PCI controller performs an internal system reset of the S_RESET type and then proceeds into a power savings mode. (The reset operation caused by $\overline{SLEEP}$ assertion will not affect BCR registers.) The PCI interface section is not effected by $\overline{SLEEP}$. In particular, access to the PCI configuration space remains possible. None of the

configuration registers will be reset by $\overline{\text{SLEEP}}$. All I/O accesses to the PCnet-PCI controller will result in a PCI target abort response. The PCnet-PCI controller will not assert $\overline{\text{REQ}}$ while in sleep mode. When $\overline{\text{SLEEP}}$ is asserted, all non-PCI interface outputs will be placed in their normal S_RESET condition. All non-PCI interface inputs will be ignored except for the $\overline{\text{SLEEP}}$ pin itself. De-assertion of $\overline{\text{SLEEP}}$ results in wake-up. The system must refrain from starting the network operations of the PCnet-PCI device for 0.5 seconds following the deassertion of the $\overline{\text{SLEEP}}$ signal in order to allow internal analog circuits to stabilize.

Both CLK and XTAL1 inputs must have valid clock signals present in order for the $\overline{\text{SLEEP}}$ command to take effect. If $\overline{\text{SLEEP}}$ is asserted while $\overline{\text{REQ}}$ is asserted, then the PCnet-PCI controller will wait for the assertion of $\overline{\text{GNT}}$. When $\overline{\text{GNT}}$ is asserted, the $\overline{\text{REQ}}$ signal will be deasserted and then the PCnet-PCI controller will proceed to the power savings mode.

The $\overline{\text{SLEEP}}$ pin should not be asserted during power supply ramp-up. If it is desired that $\overline{\text{SLEEP}}$ be asserted at power up time, then the system must delay the assertion of $\overline{\text{SLEEP}}$ until three CLK cycles after the completion of a valid pin $\overline{\text{RST}}$ operation.

XTAL1 2
Crystal Oscillator Inputs
Input/Output

The crystal frequency determines the network data rate. The PCnet-PCI controller supports the use of quartz crystals to generate a 20 MHz frequency compatible with the ISO 8802-3 (IEEE/ANSI 802.3) network frequency tolerance and jitter specifications. See the section External Crystal Characteristics (in section Manchester Encoder/Decoder) for more detail.

The network data rate is one-half of the crystal frequency. XTAL1 may alternatively be driven using an external CMOS level source, in which case XTAL2 must be left unconnected. Note that when the PCnet-PCI controller is in comma mode, there is an internal 22 KΩ resistor from XTAL1 to ground. If an external source drives XTAL1, some power will be consumed driving this resistor. If XTAL1 is driven LOW at this time power consumption will be minimized. In this case, XTAL1 must remain active for at least 30 cycles after the assertion of $\overline{\text{SLEEP}}$ and deassertion of $\overline{\text{REQ}}$.

Microwire EEPROM Interface

EESK
EEPROM Serial clock
Input/Output

The EESK signal is used to access the external ISO 8802-3 (IEEE/ANSI 802.3) address PROM. This pin is designed to directly interface to a serial EEPROM that uses the Microwire interface protocol. EESK is connected to the Microwire EEPROMs Clock pin. It is controlled by either the PCnet-PCI controller directly during a read of the entire EEPROM, or indirectly by the host system by writing to BCR19, bit 1.

The EESK pin is also used during EEPROM Auto-detection to determine whether or not an EEPROM is present at the PCnet-PCI controller Microwire interface. At the trailing edge of the $\overline{\text{RST}}$ signal, $\overline{\text{LED1}}$ is sampled to determine the value of the EEDET bit in BCR19. A sampled HIGH value means that an EEPROM is present, and EEDET will be set to ONE. A sampled LOW value means that an EEPROM is not present, and EEDET will be set to ZERO. See the EEPROM Auto-detection section for more details.

EESK is shared with the LED1 function. If no LED circuit is to be attached to this pin, then a pull up or pull down resistor must be attached instead, in order to resolve the EEDET setting.

EEDO
EEPROM Data Out
Input

The EEDO signal is used to access the external ISO 8802-3 (IEEE/ANSI 802.3) address PROM. This pin is designed to directly interface to a serial EEPROM that uses the Microwire interface protocol. EEDO is connected to the Microwire EEPROMs Data Output pin. It is controlled by the EEPROM during reads. It may be read by the host system by reading BCR19 bit 0.

EEDO is shared with the LED3 function.

EECS
EEPROM Chip Select
Output

The function of the EECS signal is to indicate to the Microwire EEPROM device that it is being accessed. The EECS signal is active high. It is controlled by either the PCnet-PCI controller during command portions of a read of the entire EEPROM, or indirectly by the host system by writing to BCR19 bit 2.

EEDI
EEPROM Data In
Output

The EEDI signal is used to access the external ISO 8802-3 (IEEE/ANSI 802.3) address PROM. EEDI functions as an output. This pin is designed to directly interface to a serial EEPROM that uses the Microwire interface protocol. EEDI is connected to the Microwire EEPROMs Data Input pin. It is controlled by either the PCnet-PCI controller during command portions of a read of the entire EEPROM, or indirectly by the host system by writing to BCR19 bit 0.

EEDI is shared with the LNKST function.

Attachment Unit Interface

CI±
Collision In
Input

A differential input pair signaling the PCnet-PCI controller that a collision has been detected on the network media, indicated by the CI± inputs being driven with a 10 MHz pattern of sufficient amplitude and pulse width to meet ISO 8802-3 (IEEE/ANSI 802.3) standards. Operates at pseudo ECL levels.

DI±
Data In
Input

A differential input pair to the PCnet-PCI controller carrying Manchester encoded data from the network. Operates at pseudo ECL levels.

DO±
Data Out
Output

A differential output pair from the PCnet-PCI controller for transmitting Manchester encoded data to the network. Operates at pseudo ECL levels.

Twisted-Pair Interface

RXD±
10BASE-T Receive Data
Input

10BASE-T port differential receivers.

TXD±
10BASE-T Transmit Data
Output

10BASE-T port differential drivers.

TXP±
10BASE-T Pre-Distortion Control
Output

These outputs provide transmit pre-distortion control in conjunction with the 10BASE-T port differential drivers.

Test Interface

NOUT
NAND Tree Out
Output

The results of the NAND tree testing can be observed on the NOUT pin. NOUT will be constantly high, when $\overline{RST}$ is deasserted.

Power Supply Pins

Analog Power Supply Pins

AV_{DD}
Analog Power (4 Pins)
Power

There are four analog +5 V supply pins. Special attention should be paid to the printed circuit board layout to avoid excessive noise on these lines. Refer to Appendix B and the *PCnet Family Technical Manual* (PID #18216A) for details.

AV_{SS}
Analog Ground (2 Pins)
Power

There are two analog ground pins. Special attention should be paid to the printed circuit board layout to avoid excessive noise on these lines. Refer to Appendix B and the *PCnet Family Technical Manual* (PID #18216A) for details.

Digital Power Supply Pins

V_{DD}
Digital Power (6 Pins)
Power

There are 6 power supply pins that are used by the internal digital circuitry. All V_{DD} pins must be connected to a +5 V supply.

V_{DDB}
I/O Buffer Power (4 Pins)
Power

There are 4 power supply pins that are used by the PCI bus Input/Output buffer drivers. All V_{DDB} pins must be connected to a +5 V supply.

V_{SS}
Digital Ground (12 Pins)
Ground

There are 12 ground pins that are used by the internal digital circuitry.

V_{SSB}
I/O Buffer Ground (8 Pins)
Ground

There are 8 ground pins that are used by the PCI bus Input/Output buffer drivers.

BASIC FUNCTIONS

System Bus Interface Function

The PCnet-PCI controller is designed to operate as a Bus Master during normal operations. Some slave I/O accesses to the PCnet-PCI controller are required in normal operations as well. Initialization of the PCnet-PCI controller is achieved through a combination of PCI Configuration Space accesses, Bus Slave accesses, Bus Master accesses and an optional read of a serial EEPROM that is performed by the PCnet-PCI controller. The EEPROM read operation is performed through the Microwire interface. The ISO 8802-3 (IEEE/ANSI 802.3) Ethernet Address may reside within the serial EEPROM. Some PCnet-PCI controller configuration registers may also be programmed by the EEPROM read operation.

The APROM, on-chip board-configuration registers, and the Ethernet controller registers occupy 32-bytes of I/O space which can be located on a wide variety of starting addresses by modifying the Base Address Register in the PCI Configuration Space.

Software Interface

The software interface to the PCnet-PCI controller is divided into three parts. One part is the PCI configuration registers. They are used to identify the PCnet-PCI controller, and are also used to setup the configuration of the device. The setup information includes the I/O base address and the routing of the PCnet-PCI controller interrupt channel. This allows for a jumperless implementation.

The second portion of the software interface is the direct access to the I/O resources of the PCnet-PCI controller. The PCnet-PCI controller occupies 32-bytes of I/O space that must begin on a 32-byte block boundary. The I/O base address can be changed to any 32-bit quantity that begins on a 32-bit block boundary by modifying the Base Address Register in the PCI Configuration Space. The 32-byte I/O space is used by the software to program the PCnet-PCI controller operating mode, to enable and disable various features, to monitor operating status and to request particular functions to be executed by the PCnet-PCI controller.

The third portion of the software interface is the descriptor and buffer areas that are shared between the software and the PCnet-PCI controller during normal network operations. The descriptor area boundaries are set by the software and do not change during normal network operations. There is one descriptor area for receive activity and there is a separate area for transmit activity. The descriptor space contains relocatable pointers to the network packet data and it is used to transfer packet status from the PCnet-PCI controller to the software. The buffer areas are locations that hold packet data for transmission or that accept packet data that has been received.

Network Interfaces

The PCnet-PCI controller can be connected to an 802.3 network via one of two network interfaces. The Attachment Unit Interface (AUI) provides an ISO 8802-3 (IEEE/ANSI 802.3) compliant differential interface to a remote MAU or an on-board transceiver. The 10BASE-T interface provides a twisted-pair Ethernet port. While in auto-selection mode, the interface in use is determined by an auto-sensing mechanism which checks the link status on the 10BASE-T port. If there is no active link status, then the device assumes an AUI connection.

DETAILED FUNCTIONS

Bus Interface Unit (BIU)

The bus interface unit is built of several state machines that run synchronously to CLK. One bus interface unit state machine handles accesses where the PCnet-PCI controller is the bus slave, and another handles accesses where the PCnet-PCI controller is the bus master. All inputs are synchronously sampled. All outputs are synchronously generated on the rising edge of CLK.

Bus Acquisition

The PCnet-PCI microcode (in the buffer management section) will determine when a DMA transfer should be initiated. The first step in any PCnet-PCI bus master transfer is to acquire ownership of the bus. This task is handled by synchronous logic within the BIU. Bus ownership is requested with the $\overline{REQ}$ signal and ownership is granted by the arbiter through the $\overline{GNT}$ signal.

Figure 1 shows the PCnet-PCI controller bus acquisition. $\overline{GNT}$ is asserted at clock 3. The PCnet-PCI controller starts driving AD[31:00] and C/$\overline{BE}$[3:0] prior to clock 4. $\overline{FRAME}$ is asserted at clock 5 indicating a valid address and command on AD[31:00] and C/$\overline{BE}$[3:0]. ADSTEP (bit 7) in the PCI Command register is set to ONE to indicated that the PCnet-PCI controller uses address stepping. Address stepping in only used for the first address phase of a bus master period.

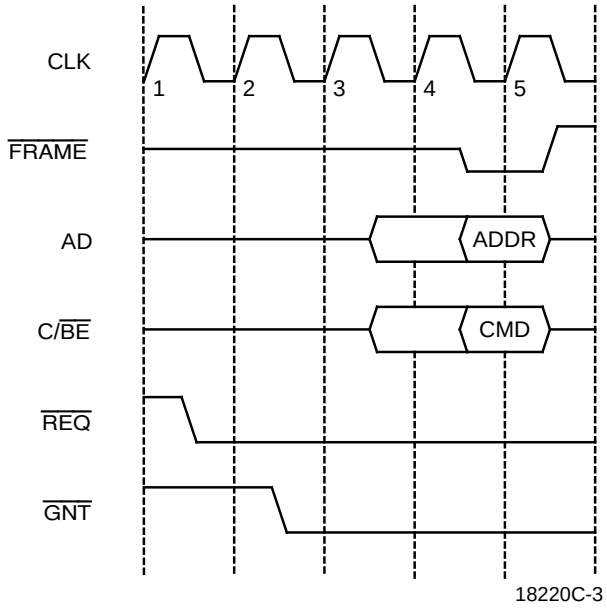


Figure 1. Bus Acquisition

Note that assertion of the STOP bit in CSR0 will not cause a deassertion of the $\overline{REQ}$ signal. Note also that a read of the RESET register, (I/O resource at offset 14h from the PCnet-PCI I/O base address) will not cause a deassertion of the $\overline{REQ}$ signal. Either of these actions will cause the internal master state machine logic to cease operations, but the $\overline{REQ}$ signal will remain active until the $\overline{GNT}$ signal is asserted. Following either of the above actions, on the next clock cycle after the $\overline{GNT}$ signal is asserted, the PCnet-PCI controller will deassert the $\overline{REQ}$ signal.

Assertion of a minimum-width pulse on the $\overline{RST}$ pin will cause the $\overline{REQ}$ signal to deassert immediately following the assertion of the $\overline{RST}$ pin. In this case, the PCnet-PCI controller will not wait for the assertion of the $\overline{GNT}$ signal before deasserting the $\overline{REQ}$ signal.

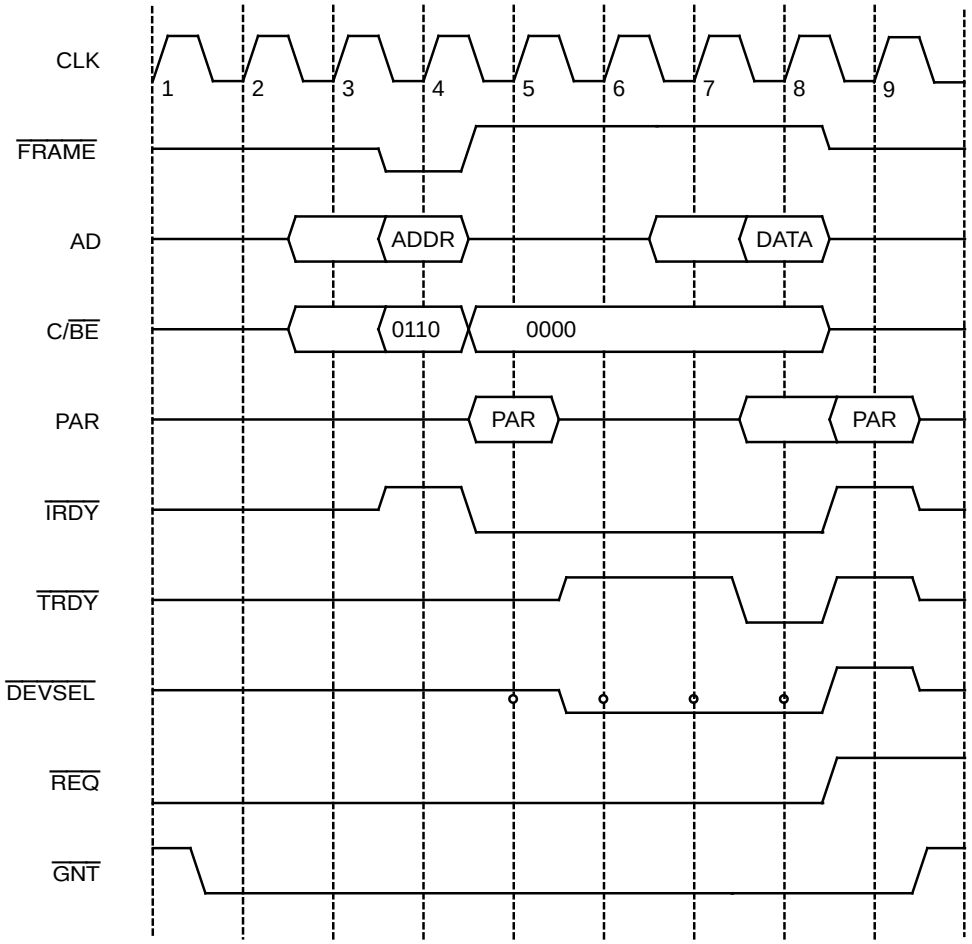
Bus Master DMA Transfers

There are four primary types of DMA transfers. The PCnet-PCI controller uses non-burst as well as burst cycles for read and write access to the main memory.

Basic Non-Burst Read Cycles

The PCnet-PCI controller uses non-burst read cycles to access the initialization block and the receive and transmit descriptor entries. Some of the read accesses to the transmit buffer memory are also in non-burst mode. All PCnet-PCI controller non-burst read accesses are of the PCI command type Memory Read (type 6). Note that during all non-burst read operations, the PCnet-PCI controller will always activate all byte enables, even though some byte lanes may not contain valid data as indicated by a buffer pointer value. In such instances, the PCnet-PCI controller will internally discard unneeded bytes.

Figure 2 shows a typical non-burst read access. The PCnet-PCI controller asserts $\overline{IRDY}$ at clock 5 immediately after the address phase and starts sampling $\overline{DEVSEL}$. The target extends the cycle by asserting $\overline{DEVSEL}$ not until clock 6. Additionally, the target inserts one wait state by asserting its ready ($\overline{TRDY}$) at clock 8.

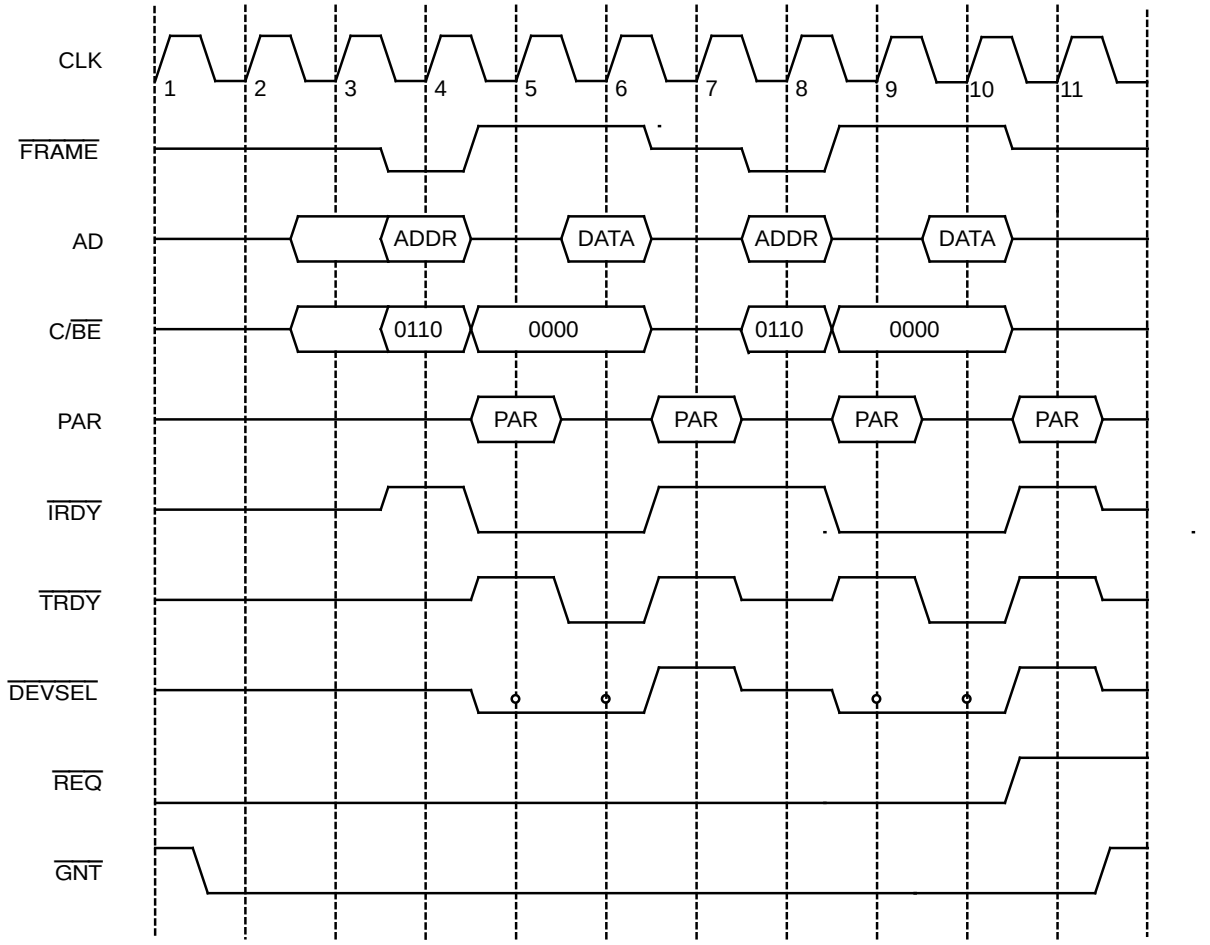


∘ $\overline{DEVSEL}$ *is sampled by the PCnet-PCI controller.*

18220C-4

Figure 2. Non-Burst Read Cycle With Wait States

Figure 3 shows two non-burst read access within one arbitration cycle. The PCnet-PCI controller will drop FRAME between two consecutive non-burst read cycles. The PCnet-PCI controller will re-request the bus right again if it is preempted before starting the second access. The example below also shows a target that can respond to the PCnet-PCI controller read cycles without wait states.



∘ DEVSEL *is sampled by the PCnet-PCI controller.*

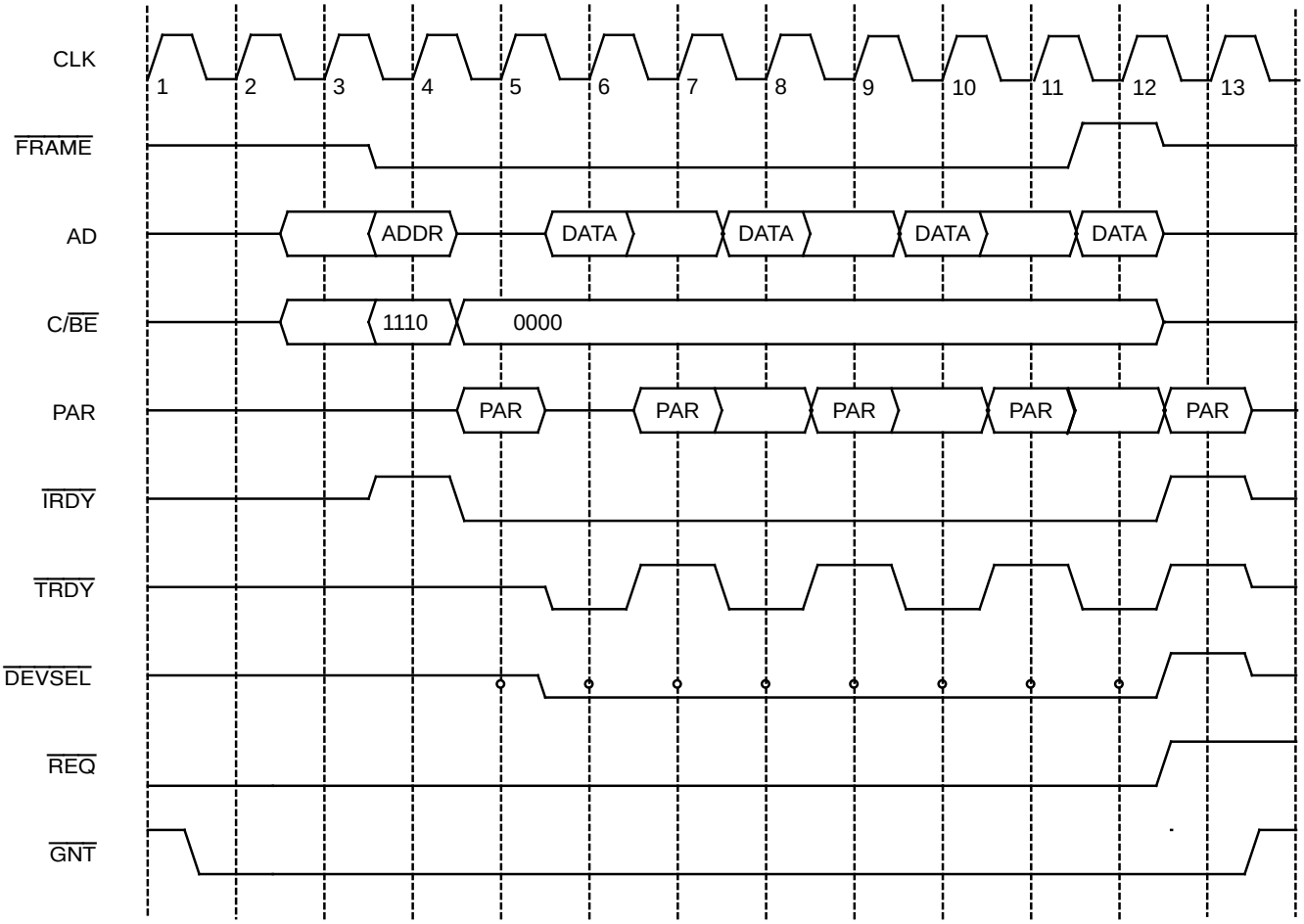
18220C-5

Figure 3. Non-Burst Read Cycles Without Wait States

Basic Burst Read Cycles

The PCnet-PCI controller provides a burst mode to read data from the transmit buffer. The burst mode must be enabled by setting BREADE in BCR18. All PCnet-PCI controller burst read transfers are of the PCI command type Memory Read Line (type15). AD[1:0] will both be ZERO during the address phase indicating a linear burst order. All four byte enable signals will be ZERO during the data phase as the PCnet-PCI controller always reads a full 32-bit word when in burst mode.

Figure 4 shows a typical burst read access. The PCnet-PCI controller arbitrates for the bus, is granted access, and reads four 32-bit words (DWORD) from system memory and then releases the bus. All four data phases in this example take two clock cycles each, which is determined by the timing of $\overline{\text{TRDY}}$.



∘ $\overline{\text{DEVSEL}}$ *is sampled by the PCnet-PCI controller.*

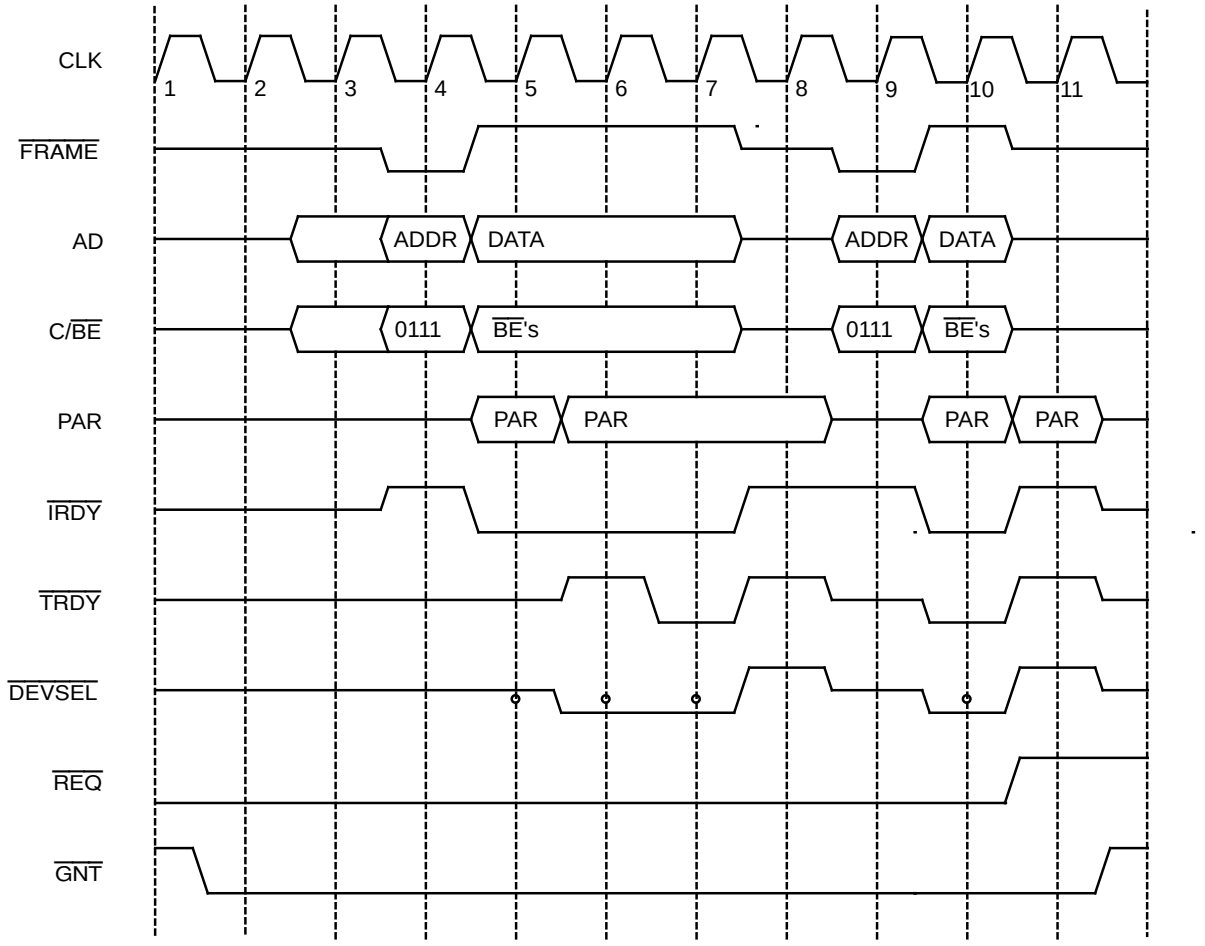
18220C-6

Figure 4. Burst Read Cycles

Basic Non-Burst Write

The PCnet-PCI controller uses non-burst write cycles to access the receive and transmit descriptor entries. Some of the write accesses to the receive buffer memory are also in non-burst mode. All PCnet-PCI controller non-burst write accesses are of the PCI command type Memory Write (type 7).

Figure 5 shows two non-burst write access within one arbitration cycle. The PCnet-PCI controller will drop $\overline{\text{FRAME}}$ between two consecutive non-burst write cycles. The PCnet-PCI controller will re-request the bus immediately if it is preempted before starting the second access. The example below shows an extended cycle for the first access. The target asserts $\overline{\text{DEVSEL}}$ 2 clock cycles after the address phase ($\overline{\text{FRAME}}$ asserted) and adds one extra wait state by asserting $\overline{\text{TRDY}}$ only on clock 7. The second write cycle in the example shows a ZERO wait state access.



∘ $\overline{\text{DEVSEL}}$ *is sampled by the PCnet-PCI controller.*

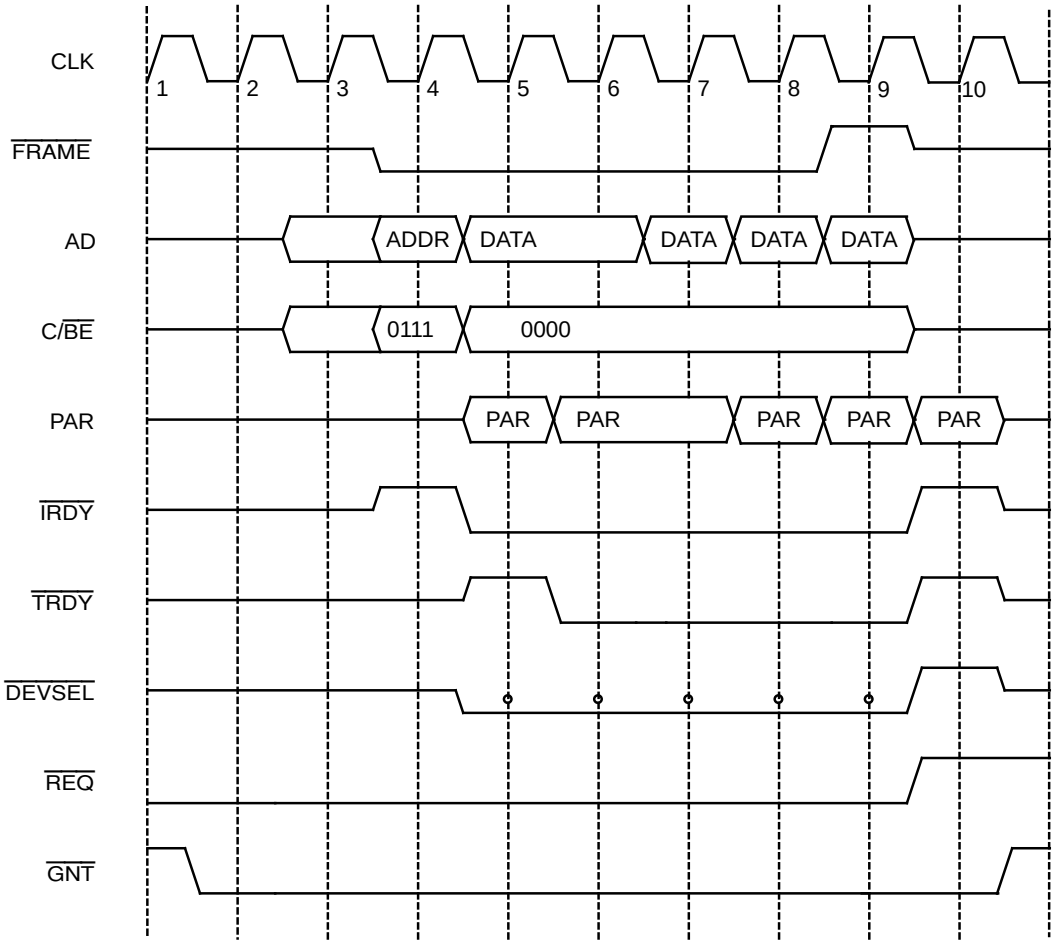
18220C-7

Figure 5. Non-Burst Write Cycles With and Without Wait States

Basic Burst Write Cycles

The PCnet-PCI controller provides a burst mode to write data to the receive buffer. The burst mode must be enabled by setting BWRITE in BCR18. All PCnet-PCI controller burst write transfers are of the PCI command type Memory Write (type 7). AD[1:0] will both be ZERO during the address phase indicating a linear burst order. All four byte enable signals will be ZERO during the data phase as the PCnet-PCI controller always writes a full 32-bit word when in burst mode.

Figure 6 shows a typical burst write access. The PCnet-PCI controller arbitrates for the bus, is granted access, and writes four 32-bit words (DWORDs) from system memory and then releases the bus. In this example, the memory system extends the data phase of the first access by one wait state. The following three data phases take one clock cycle each, which is determined by the timing of $\overline{\text{TRDY}}$.



o $\overline{\text{DEVSEL}}$ *is sampled by the PCnet-PCI controller.*

18220C-8

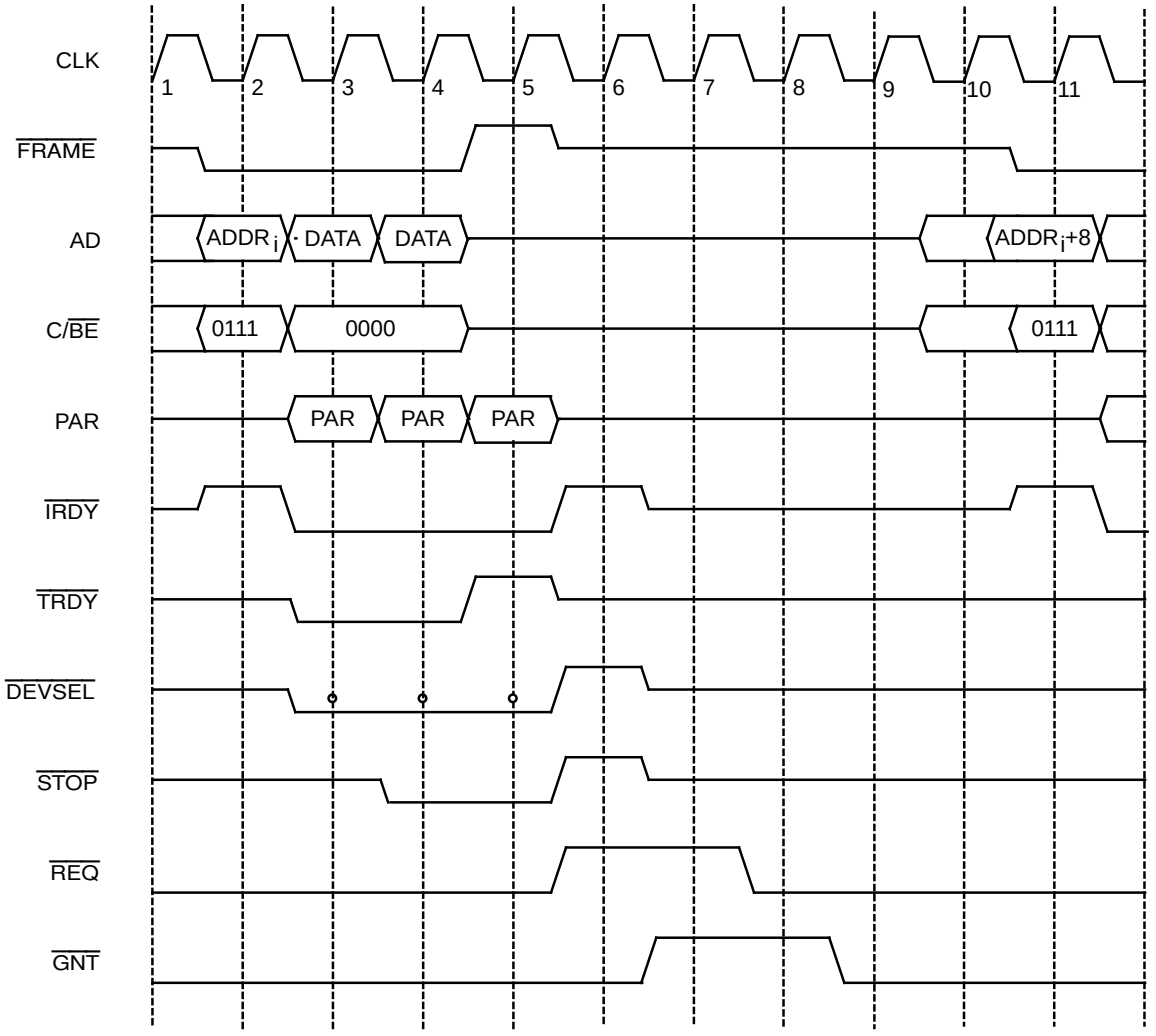
Figure 6. Burst Write Cycles

Target Initiated Termination

When the PCnet-PCI controller is a bus master, the cycles it produces on the PCI bus may be terminated by the target in one of three different ways.

Disconnect With Data Transfer

Figure 7 shows a disconnection in which one last data transfer occurs after the target asserted $\overline{STOP}$. $\overline{STOP}$ is asserted on clock 4 to start the termination sequence. Data is still transferred during this cycles, since both $\overline{IRDY}$ and $\overline{TRDY}$ are asserted. The PCnet-PCI controller terminates the current transfer with the deassertion of $\overline{FRAME}$ on clock 5 and then one clock cycle later with the deassertion $\overline{IRDY}$. It finally releases the bus on clock 6. The PCnet-PCI controller will re-request the bus after 2 clock cycles, if it wants to transfer more data. The starting address of the new transfer will be the address of the next untransferred data.



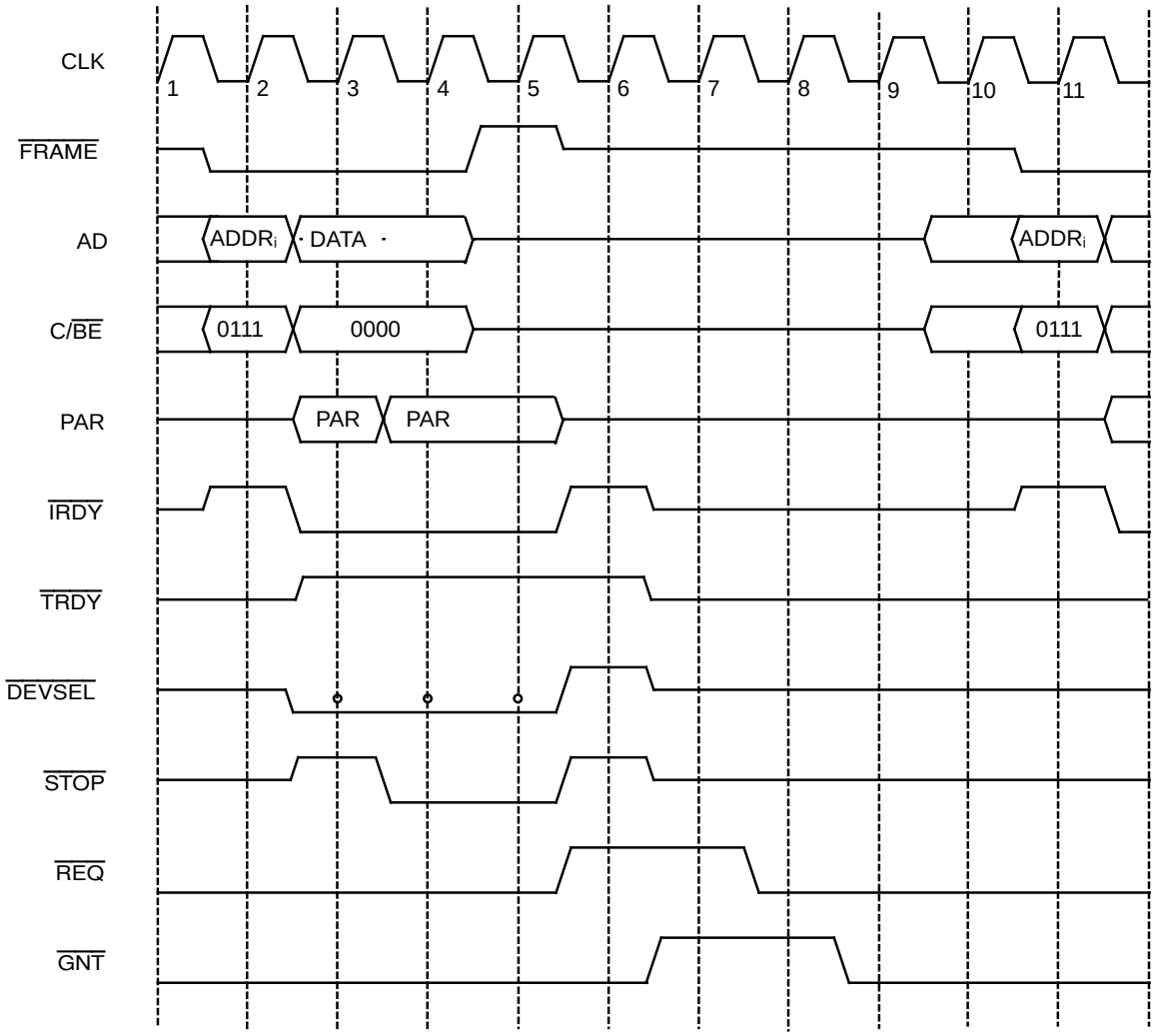
∘ $\overline{DEVSEL}$ *is sampled by the PCnet-PCI controller.*

18220C-9

Figure 7. Disconnect with Data Transfer

Disconnect Without Data Transfer

Figure 8 shows a target disconnect sequence during which no data is transferred. $\overline{STOP}$ is asserted on clock 4 without $\overline{TRDY}$ being asserted at the same time. The PCnet-PCI controller terminates the current transfer with the deassertion of $\overline{FRAME}$ on clock 5 and one clock cycle later with the deassertion of $\overline{IRDY}$. It finally releases the bus on clock 6. The PCnet-PCI controller will re-request the bus after 2 clock cycles to retry the last transfer. The starting address of the new transfer will be the same address as of the last untransferred data.



∘ $\overline{DEVSEL}$ *is sampled by the PCnet-PCI controller.*

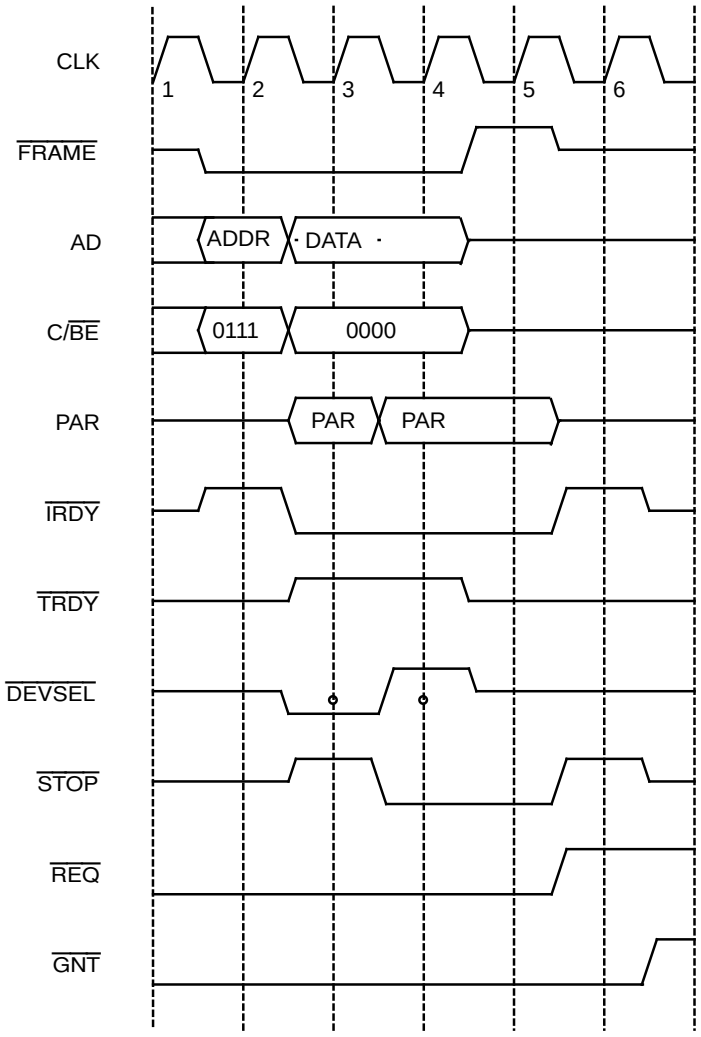
18220C-10

Figure 8. Disconnect Without Data Transfer

Target Abort

Figure 9 shows a target abort sequence. The target asserts DEVSEL for one clock. It then deasserts DEVSEL and asserts STOP on clock 4. A target can use the target abort sequence to indicate that it cannot service the data transfer and that it does not want the transaction to be retried. Additionally, the PCnet-PCI controller cannot make any assumption about the success of the previous data transfers in the current transaction. The PCnet-PCI controller terminates the current transfer with the deassertion of FRAME on clock 5 and one clock cycle later with the deassertion of IRDY. It finally releases the bus on clock 6.

Since data integrity is not guaranteed, the PCnet-PCI controller cannot recover from a target abort event. The PCnet-PCI controller will reset all CSR and BCR locations to their H_RESET values. Any on-going network activity will be stopped immediately. The PCI configuration registers will not be cleared. RTABORT (bit 12) in the Status register will be set to indicate that the PCnet-PCI controller has received a target abort.



o DEVSEL *is sampled by the PCnet-PCI controller.*

18220C-11

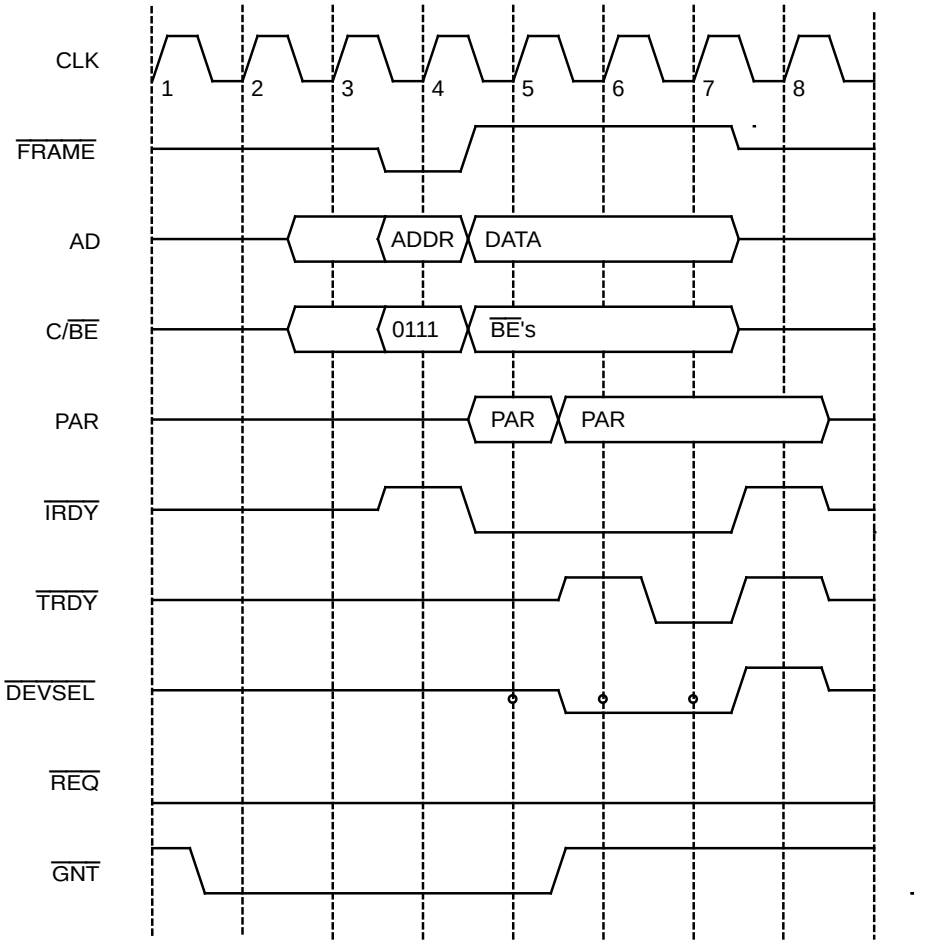
Figure 9. Target Abort

Master Initiated Termination

There are three scenarios besides normal completion of a transaction where the PCnet-PCI controller will terminate the cycles it produces on the PCI bus.

Preemption When FRAME is Deasserted

The PCnet-PCI controller will generate multiple address phases during a single bus ownership period when it is accessing the initialization block, the descriptor ring entries and the data buffers in main memory. FRAME is deasserted in between two address phases. While FRAME is deasserted, the central arbiter can remove GNT to the PCnet-PCI controller at any time to service another master. When GNT is removed, the PCnet-PCI controller will finish the current transfer and then release the bus. It will keep REQ asserted to regain bus ownership as soon as possible.



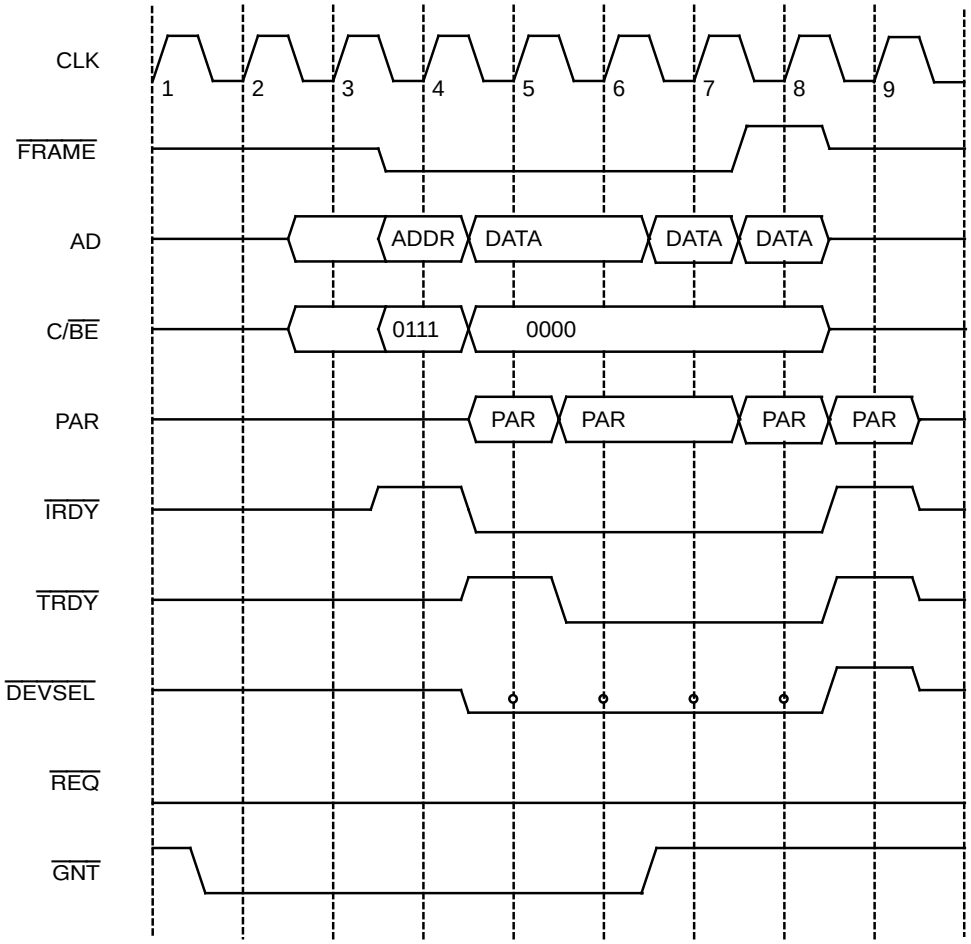
o DEVSEL *is sampled by the PCnet-PCI controller.*

18220C-12

Figure 10. Preemption When FRAME is Deasserted

Preemption When $\overline{\text{FRAME}}$ is Asserted

The central arbiter can take $\overline{\text{GNT}}$ to the PCnet-PCI controller away if the current bus operation takes too long. This may happen e.g. when the PCnet-PCI controller tries to fill the whole transmit FIFO and the target inserts extra wait states for every data phase. When $\overline{\text{GNT}}$ is taken away, the PCnet-PCI controller will finish the current transfer and then immediately release the bus. The Latency Timer in PCI configuration space of the PCnet-PCI controller is always set to ZERO. The PCnet-PCI controller will keep $\overline{\text{REQ}}$ asserted to regain bus ownership as soon as possible.



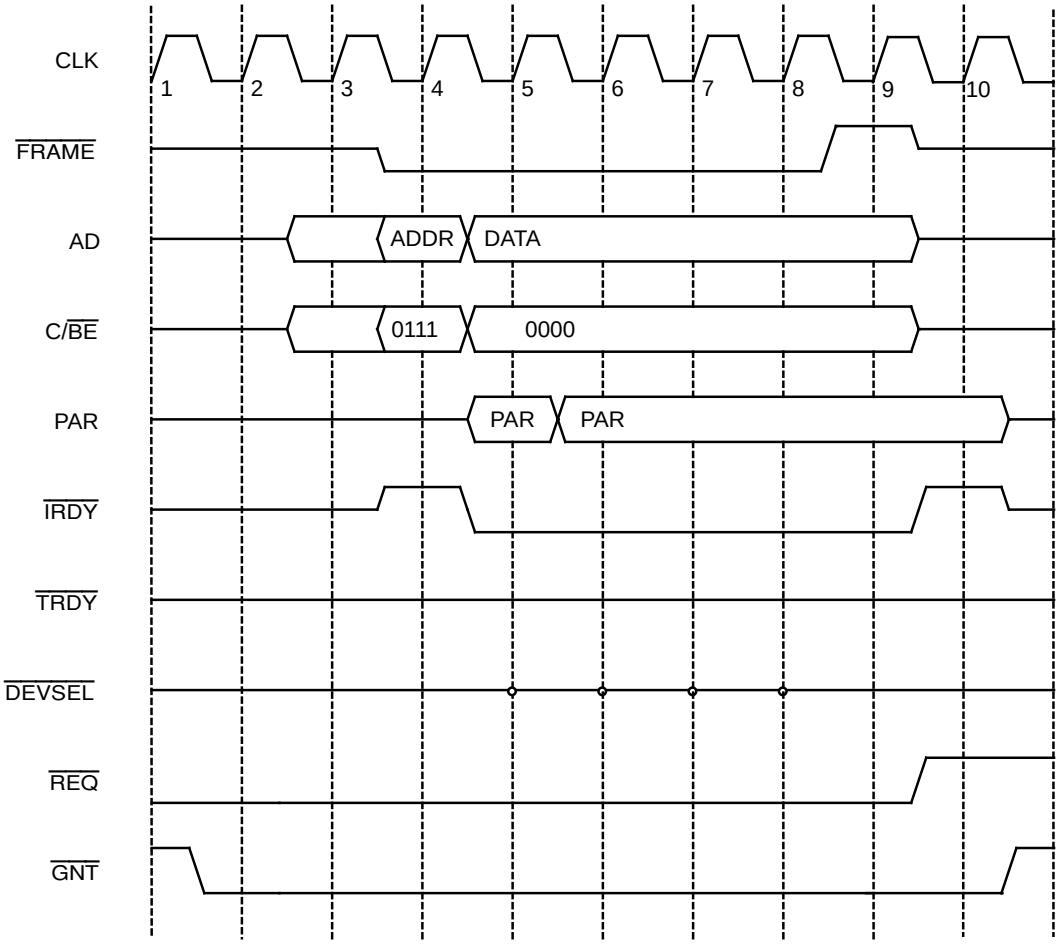
∘ $\overline{\text{DEVSEL}}$ *is sampled by the PCnet-PCI controller.*

18220C-13

Figure 11. Preemption When $\overline{\text{FRAME}}$ is Asserted

Master Abort

The PCnet-PCI controller will terminate its cycle with a Master Abort sequence if DEVSEL is not asserted within 4 clocks after FRAME is asserted. Master Abort is treated as a fatal error by the PCnet-PCI controller. The PCnet-PCI controller will reset all CSR and BCR locations to their H_RESET values. Any on-going network activity will be stopped immediately. The PCI configuration registers will not be cleared. RMABORT (bit 13) in the Status register will be set to indicate that the PCnet-PCI controller has terminated its transaction with a master abort.



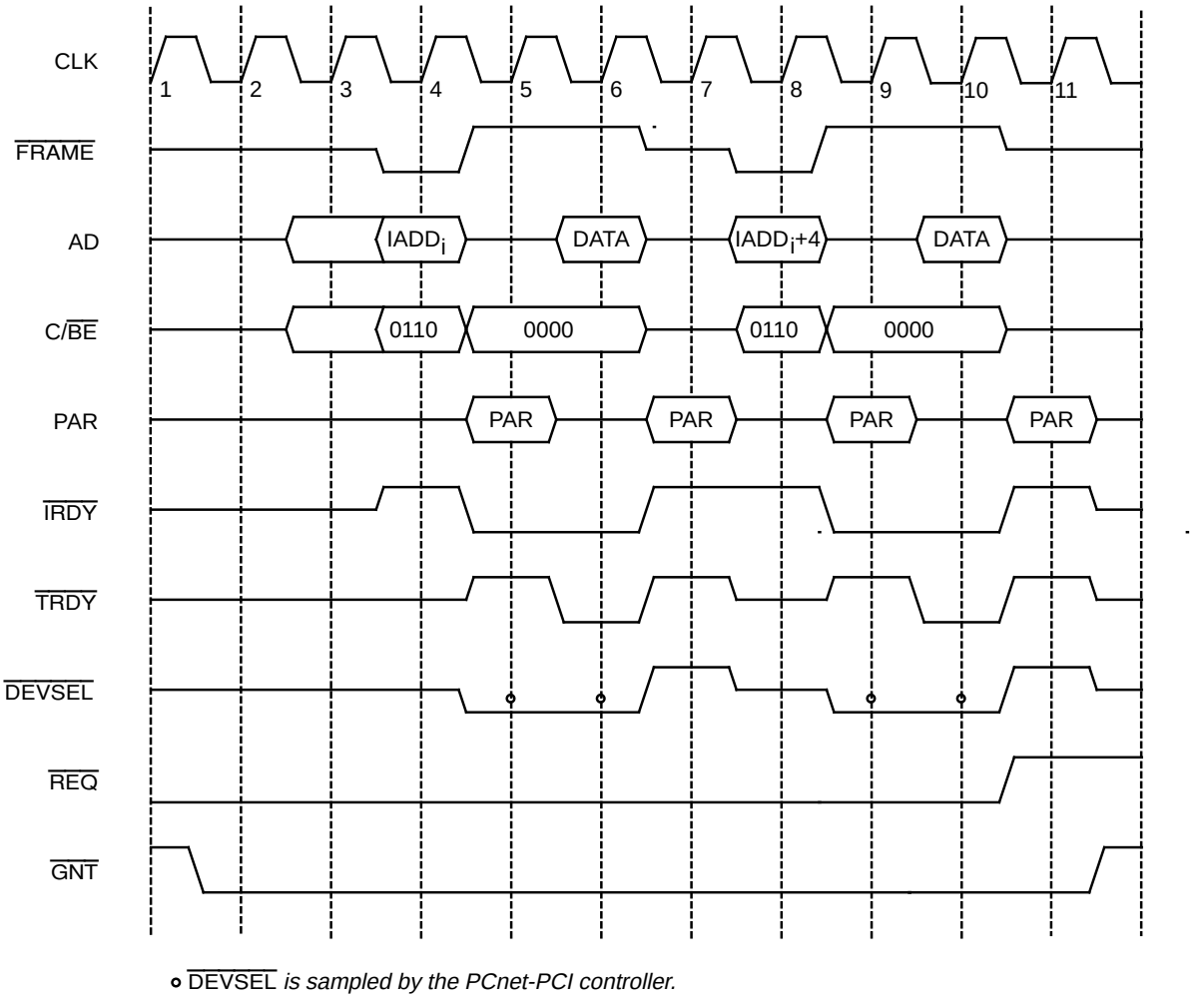
o DEVSEL *is sampled by the PCnet-PCI controller.*

18220C-14

Figure 12. Master Abort

Initialization Block DMA Transfers

During execution of the PCnet-PCI controller bus master initialization procedure, the PCnet-PCI microcode will repeatedly request DMA transfers from the BIU. During each of these initialization block DMA transfers, the BIU will perform two data transfer cycles (eight bytes) and then it will relinquish the bus. The two transfers within the mastership period will always be read cycles to ascending contiguous addresses. When SSIZE32 = 1 (BCR20, bit 8), there are 7 DWORDs to transfer during the bus master initialization procedure, so four bus mastership periods are needed in order to complete the initialization sequence. Note that the last DWORD transfer of the last bus mastership period of the initialization sequence accesses an unneeded location. Data from this transfer is discarded internally. When SSIZE32 = 0 (BCR20, bit 8), then three bus mastership periods are needed to complete the initialization sequence.



∘ DEVSEL *is sampled by the PCnet-PCI controller.*

18220C-15

Figure 13. Initialization Block Read

Descriptor DMA Transfers

PCnet-PCI microcode will determine when a descriptor access is required. A descriptor DMA read will consist of two DWORD (double-word) transfers. A descriptor DMA write will consist of one or two DWORD transfers. (The transfers within a descriptor DMA transfer mastership period will always be of the same type (either all read or all write)).

If buffer chaining is used, writes to the descriptors of all intermediate buffers consist of only one DWORD to return OWNership of the buffer to the system. On all single buffer transmit or receive descriptors, as well as on the last buffer in chain, writes to the descriptor consist of two DWORDs. The first DWORD containing status information. The second DWORD containing additional status and the OWNership bit (i.e. MD1[31]).

The transfers will be addressed as specified in tables 1 and 2.

Table 1. Bus Master Reads of Descriptors

16-Bit Software Mode			32-Bit Software Mode		
Address Sequence AD[7:0]*	**LANCE/ PCnet-ISA Item Accessed**	**PCnet-PCI Item Accessed**	**Address Sequence AD[7:0]***	**LANCE/ PCnet-ISA Item Accessed**	**PCnet-PCI Item Accessed**
00	MD1[15:0], MD0[15:0]	MD1[31:24], MD0[23:0]	04	MD1[15:8], MD2[15:0]	MD1[31:0]
04	MD3[15:0], MD2[15:0]	MD2[15:0], MD1[15:0]	00	MD1[7:0], MD0[15:0]	MD0[31:0]
Bus Break			Bus Break		

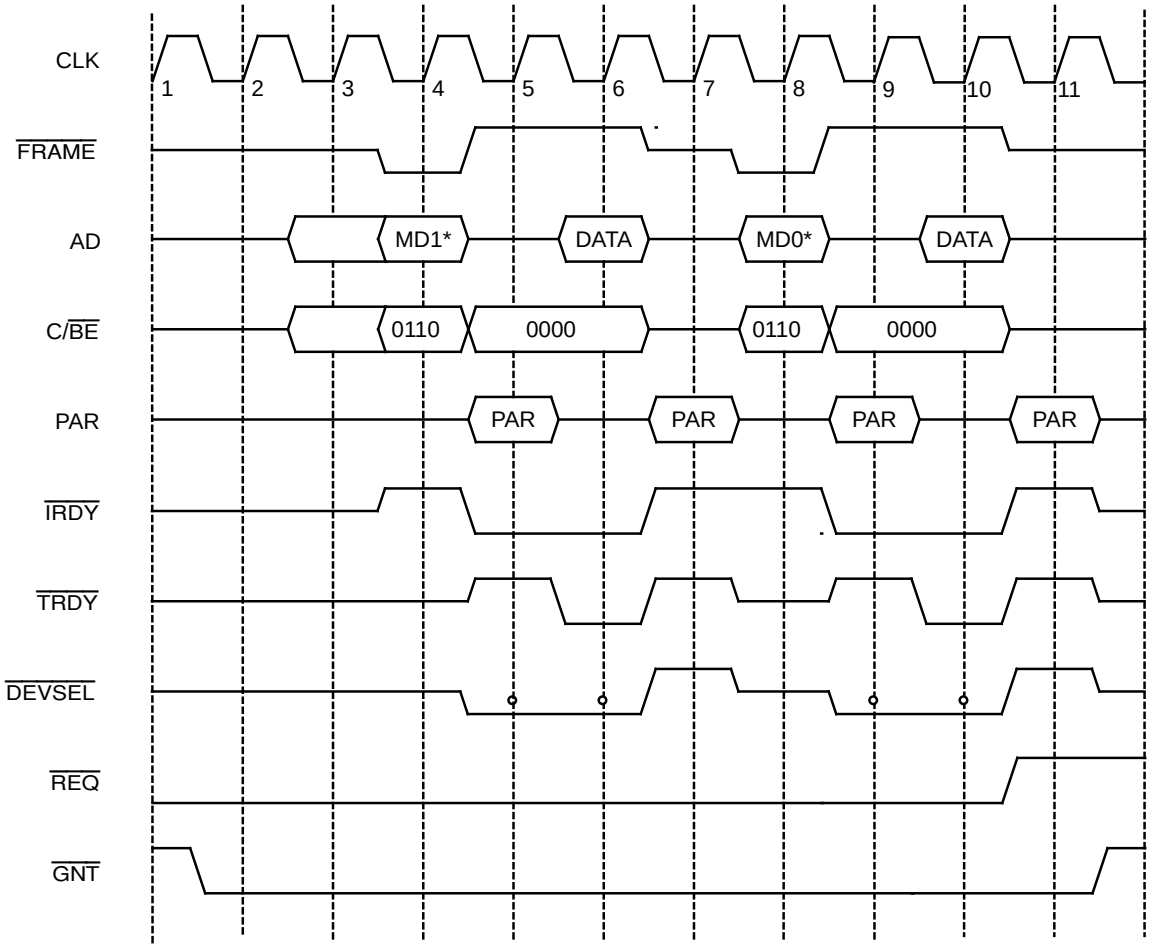
Table 2. Bus Master Writes to Descriptors

16-Bit Software Mode			32-Bit Software Mode		
Address Sequence AD[7:0]*	**LANCE/ PCnet-ISA Item Accessed**	**PCnet-PCI Item Accessed**	**Address Sequence AD[7:0]***	**LANCE/ PCnet-ISA Item Accessed**	**PCnet-PCI Item Accessed**
04	MD3[15:0], MD2[15:0]	MD2[15:0], MD1[15:0]	08	MD3[15:0]	MD2[31:0]
00	MD1[15:0], MD0[15:0]	MD1[31:24], MD0[23:0]	04	MD1[15:8], MD2[15:0]	MD1[31:0]
Bus Break			Bus Break		

** Address values for AD[31:08] are constant throughout any single descriptor DMA transfer. AD[1:0] must be set to ZERO in the descriptor base address.*

During descriptor read accesses, the byte enable signals will indicate that all byte lanes are active. Should some of the bytes not be needed, then the PCnet-PCI controller will internally discard the extraneous information that was gathered during such a read. During write accesses, only the bytes which need to be written are enabled, by activating the corresponding byte enable pins.

The only significant differences between descriptor DMA transfers and initialization DMA transfers are that the addresses of the accesses follow different ordering.

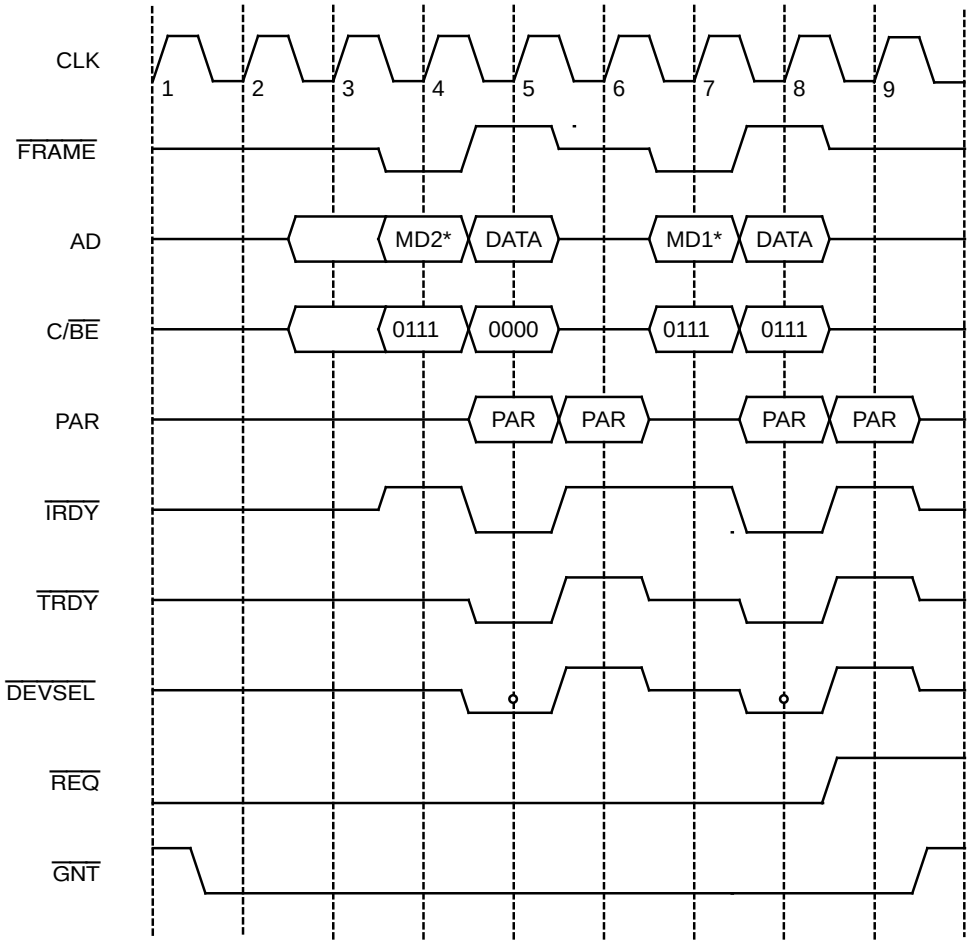


∘ DEVSEL *is sampled by the PCnet-PCI controller.*

** Note that Message Descriptor addresses 1 and 0 are in descending order.*

18220C-16

Figure 14. Descriptor Ring Read



∘ DEVSEL *is sampled by the PCnet-PCI controller.*

** Note that Message Descriptor addresses 2 and 1 are in descending order.*

18220C-17

Figure 15. Descriptor Ring Write

FIFO DMA Transfers

PCnet-PCI microcode will determine when a FIFO DMA transfer is required. This transfer mode will be used for transfers of data to and from the PCnet-PCI FIFOs. Once the PCnet-PCI BIU has been granted bus mastership, it will perform a series of consecutive transfer cycles before relinquishing the bus. All transfers within the master cycle will be either read or write cycles, and all transfers will be to contiguous, ascending addresses. Both non-burst and burst cycles are used.

Non-Burst FIFO DMA Transfers

Non-burst FIFO DMA transfers is the default mode the PCnet-PCI controller uses to read and write data when accessing the FIFOs. Each non-burst transfer will be performed sequentially, with the issue of an address, and the transfer of the corresponding data with appropriate output signals to indicate selection of the active data bytes during the transfer. $\overline{\text{FRAME}}$ will be dropped after every address phase. (See figures 2,3 and 5.) The number of data transfer cycles contained within a single bus mastership period is in general dependent on the programming of the DMAPLUS option (CSR4, bit 14). Several other factors will also affect the length of the bus mastership period. The possibilities are as follows:

- If DMAPLUS = 0, a maximum of 16 transfers will be performed by default. This default value may be changed by writing to the DMA Transfer Counter (CSR80). Note that DMAPLUS = 0 merely sets a maximum value. The minimum number of transfers in the bus mastership period will be determined by all of the following variables: the settings of the FIFO watermarks and the conditions of the FIFOs, the value of the DMA Transfer Counter (CSR80), the value of the DMA Bus Timer (CSR82), and any occurrence of preemption that takes place during the bus mastership period.
- If DMAPLUS = 1, the bus cycle will continue until the transmit FIFO is filled to its high threshold (read transfers) or the receive FIFO is emptied to its low threshold (write transfers), or until the DMA Bus Timer value (CSR82) has expired. Other variables may also affect the end point of the bus mastership period in this mode. Among those variables are the particular conditions existing within the FIFOs, receive and transmit status conditions, and bus preemption events.

The FIFO thresholds are programmable (see description of CSR80), as are the DMA Transfer Counter and Bus Timer values. The exact number of transfer cycles in the case of DMAPLUS = 1 will be dependent on the latency of the system bus to the PCnet-PCI controller's mastership request and the speed of bus operation, but will be limited by the value in the Bus Timer register, the FIFO condition, receive and transmit status, and by preemption events. Barring a time-out by either of these registers, or a bus preemption by another mastering device, or exceptional receive and transmit events, or an end of packet signal from the FIFO, the FIFO watermark settings and the extent of Bus Grant latency will be the major factors determining the number of accesses performed during any given arbitration cycle when DMAPLUS = 1.

The $\overline{\text{TRDY}}$ response of the memory device will also affect the number of transfers when DMAPLUS = 1, since the speed of the accesses will affect the state of the FIFO. (During accesses, the FIFO may be filling or emptying on the network end. A slower memory response will allow additional data to accumulate inside of the FIFO (during write transfers from the receive FIFO). If the accesses are slow enough, a complete DWORD may become available before the end of the arbitration cycle and thereby increase the number of transfers in that cycle.) The general rule is that the longer the Bus Grant latency or the slower the bus transfer operations (or clock speed) or the higher the transmit watermark or the lower the receive watermark or any combination thereof the longer will be the average bus mastership period.

Burst FIFO DMA Transfers

Bursting is only performed by the PCnet-PCI controller if the BREADE and/or BWRITE bits of BCR18 are set. These bits individually enable/disable the ability of the PCnet-PCI controller to perform burst accesses during master read operations and master write operations, respectively. Only FIFO data transfers will make use of the burst mode.

The first transfer in the burst will consist of both an address and a data phase. Subsequent transfers will contain data only. AD[1:0] will always be ZERO during the address phase indicating a linear burst order. Note, that the terms 'burst' and 'linear burst' are used interchangeably throughout this document.

The number of data phases within the burst transfer is determined by the LINBC value from the BCR18 register. The burst upper limit is calculated by taking the BCR18 LINBC[2:0] value and multiplying it by 4. The result is the number of transfers that will be performed within a single linear burst sequence. When the LINBC upper limit of data transfers have been performed, a new $\overline{\text{FRAME}}$ may be asserted (if there is more data to be transferred), with a new address on the AD pins. Following the assertion of a new $\overline{\text{FRAME}}$, the linear bursting of data will resume. All byte lanes will always be active during all burst transfers as reflected by the C/$\overline{\text{BE}}$[3:0] signals.

The number of data transfer cycles within the total bus mastership period is dependent on the programming of the DMAPLUS option (CSR4, bit 14). The possibilities are as follows:

- If DMAPLUS = 0, a maximum of 16 transfers will be performed by default. This default value may be changed by writing to the DMA Transfer Counter (CSR80). Note that DMAPLUS = 0 merely sets a maximum value. The minimum number of transfers in the bus mastership period will be determined by all of the following variables: the settings of the FIFO watermarks and the conditions of the FIFOs, the value of the DMA Transfer Counter (CSR80), the value of the DMA Bus Timer (CSR82), and any occurrence of preemption that takes place during the bus mastership period.
- If DMAPLUS = 1, linear bursting will continue until the transmit FIFO is filled to its high threshold (read transfers) or the receive FIFO is emptied to its low threshold (write transfers), or until the DMA Bus Timer value (CSR82) has expired. A bus preemption event is another cause of termination of cycles. The FIFO thresholds are programmable (see description of CSR80), as are the DMA Transfer Counter and Bus Timer values. The exact number of total transfer cycles in the case of DMAPLUS = 1 will be dependent on the latency of the system bus to the PCnet-PCI controller's mastership request and the speed of bus operation, but will be limited by the value in the Bus Timer Register, the FIFO condition and by preemption occurrences, if any.

Note that the number of transfer cycles for each $\overline{\text{FRAME}}$ assertion will always only be controlled by LINBC, Bus Grant and FIFO conditions. The number of transfer cycles for each $\overline{\text{FRAME}}$ assertions will not be affected by DMAPLUS or by the values in the DMA Transfer Count register and Bus Timer register. However, these factors can influence the number of transfers that is performed during any given bus mastership period.

Barring a time-out by the DMA Transfer Count register or the Bus Timer register or a bus preemption by another mastering device, the FIFO watermark settings and the extent of Bus Grant latency will be the major factors in determining the number of accesses performed during any given bus mastership period. The $\overline{\text{TRDY}}$ response time of the memory device will also affect the number of transfers, since the speed of the accesses will affect the state of the FIFO. (During accesses, the FIFO may be filling or emptying on the network end. For example, on a Receive operation, a slower device will allow additional data to accumulate inside of the FIFO. If the accesses are slow enough, a complete DWORD may become available before the end of the bus mastership period and thereby increase the number of transfers in that period.) The general rule is that the longer the Bus Grant latency or the slower the bus transfer operations or the slower the clock speed or the higher the transmit watermark or the lower the receive watermark or any combination thereof, will produce longer total burst lengths.

Linear Burst DMA Starting Address Restrictions

A PCnet-PCI controller linear burst will begin only when the address of the current transfer meets the following condition:

AD[31:00] MOD (LINBC x 16) = 0,

The following table illustrates all possible starting address values for all legal LINBC values. Note that AD[31:06] are don't care values for all addresses. Also note that while AD[1:0] do not physically exist within a 32 bit system (the PCnet-PCI controller always drives AD[1:0] to ZERO during the address phase to indicate a linear burst order), they are valid bits within the buffer pointer field of descriptor word 0. Thus, where AD[1:0] are listed, they refer to the lowest two bits of the descriptors buffer pointer field. These bits will have an affect on determining when a PCnet-PCI controller linear burst operation may legally begin and they will affect the output values of the byte enable pins, therefore they have been included in the table as AD[1:0].

Table 3. Linear Burst DMA Starting Address Values

LINBC[2:0]	LBS = Linear Burst Size (number of transfers)	Size of Burst (bytes)	Linear Burst Beginning Addresses AD[5:0] = (Hex) (AD[31:06] = (don't care)
1	4	16	00, 10, 20, 30
2	8	32	00, 20
4	16	64	00
0, 3, 5, 7	Reserved	Reserved	Not Applicable

It is not necessary for the software to insure that the buffer address pointer contained in descriptor word 0 matches the address restrictions given in the table. If the buffer pointer does not meet the conditions set forth in the table, then the PCnet-PCI controller will simply postpone the start of linear bursting until enough non-burst FIFO DMA transfers have been performed to bring the current working buffer pointer value to a valid linear burst starting address. This operation is referred to as aligning the buffer address to a valid linear burst starting address. Once this has been done, the PCnet-PCI controller will recognize that the address for the current access is a valid linear burst starting address, and it will automatically begin to perform linear burst accesses at that time, provided of course that the software has enabled the linear burst mode.

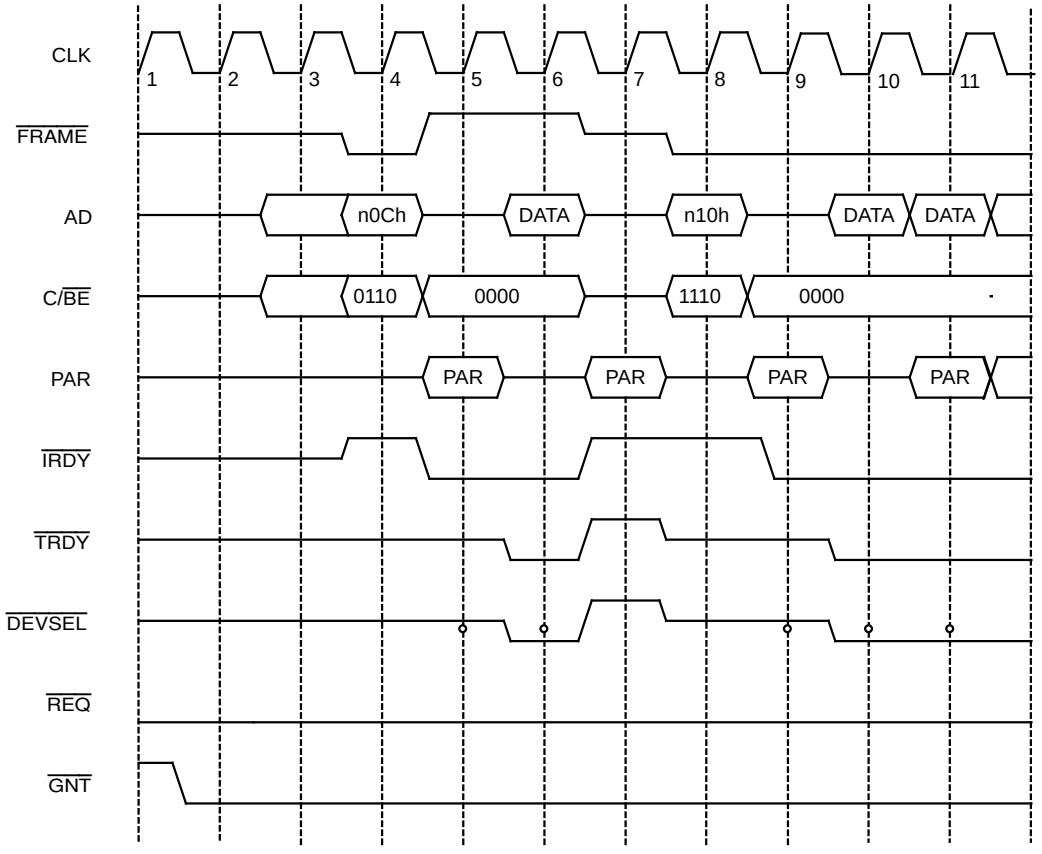
Note that if the software would provide only valid linear burst starting addresses in the buffer pointer, then the PCnet-PCI controller could avoid performing the alignment operation. It would begin linear burst accesses on the very first of the buffer transfers thereby allowing a slight gain in bus bandwidth efficiency.

Linear Burst DMA Address Alignment

Linear bursting may begin during a bus mastership period which was initially performing only non-burst operations. A change from non-burst operation to linear bursting will normally occur during linear burst DMA address alignment operations.

If the PCnet-PCI controller is programmed for burst mode (i.e. BREADE and/or BWRITE bits of BCR18 are set to ONE), and the PCnet-PCI controller requests the bus, but the starting address of the first transaction does not meet the conditions as specified in the table above, then the PCnet-PCI controller will perform non-burst accesses until it arrives at an address that does meet the conditions described in the table. At that time, and without releasing the bus, the PCnet-PCI controller will invoke the linear burst mode.

Figure 16 shows an example of a linear burst DMA alignment operation being performed. The first access to the transmit buffer is in non-burst mode, because the current address does not align with a linear burst boundary. The PCnet-PCI controller switches to burst mode beginning with the second transfer. $\overline{REQ}$ stays asserted during all transfers.



o $\overline{DEVSEL}$ *is sampled by the PCnet-PCI controller.*

18220C-18

Figure 16. Burst Alignment

Partial Linear Burst

Certain factors may cause the PCnet-PCI controller to burst fewer than the LINBC limit during a single burst sequence. Factors that could generate a partial linear burst include:

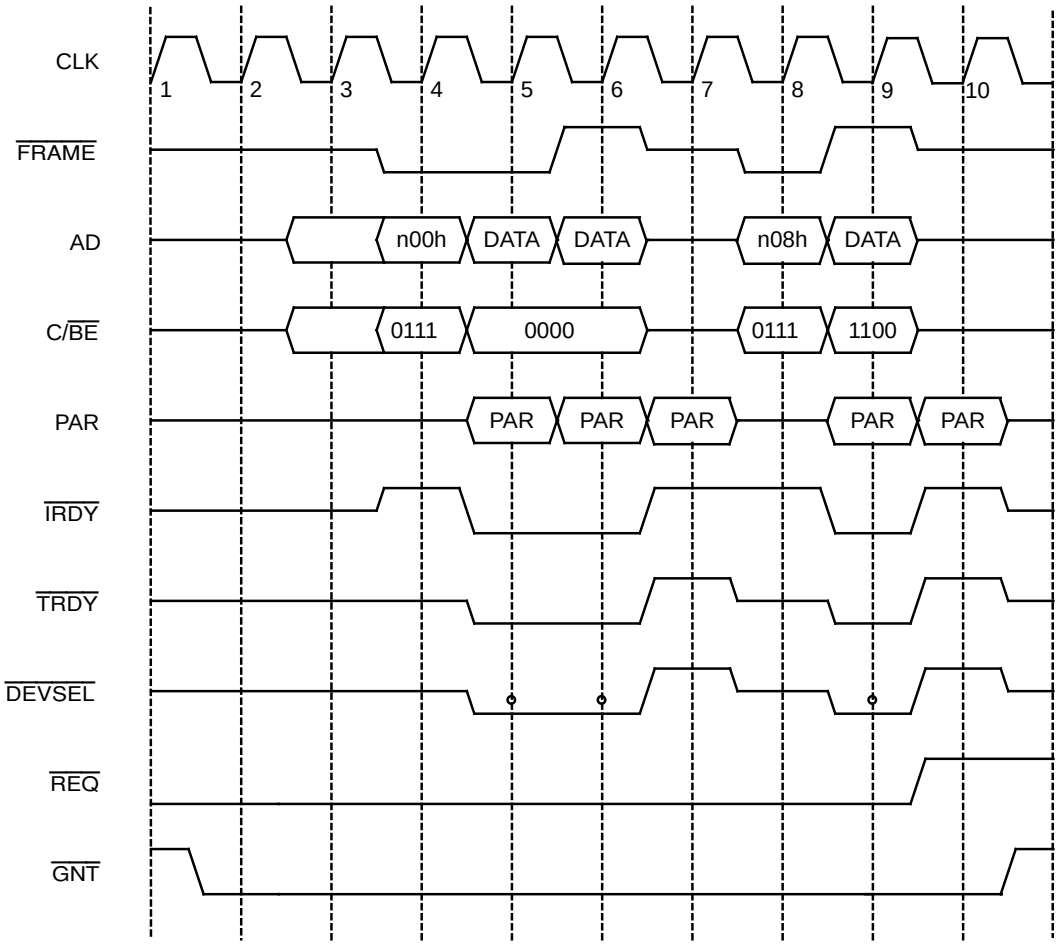
- No more data available for transfers from the current TX buffer
- No more data available for transfer from the RX FIFO for this packet
- No more space available for transfers to the current RX buffer
- Preemption

Typically, during the case of a master read operation (for TX buffer transfers), the last transfer in the linear burst sequence will be the last transfer executed before the PCnet-PCI controller releases the bus. This is true of both partial and completed linear burst sequences.

During the case of a master write operation (for RX buffer transfers) when RX packet data has ended, the last transfer in the linear burst sequence will be the last transfer executed before the PCnet-PCI controller releases the bus. This is true of both partial and completed linear burst sequences.

However, if the next transfer that the PCnet-PCI controller is scheduled to execute will be to the last available location of a RX buffer, then the PCnet-PCI controller will use a non-burst cycle to make the last transfer to the buffer. This event occurs because of the restrictions placed upon the byte enable signals during the linear burst operation. As mentioned in the initial description of linear burst accesses, all byte lanes of the data bus are always enabled during linear burst operations. Note, however, that in the case of the last RX buffer location, the PCnet-PCI controller may own only a portion of the DWORD location. In such cases, it is necessary to discontinue linear burst accesses on the second from last RX buffer location so that an ordinary transfer with some byte lanes disabled can be used for the final transfer.

Figure 17 shows a partial linear burst that occurred while approaching the transfer of the last bytes of data to a RX buffer. The linear burst begins when 10 bytes of space still remain in the RX buffer. (The number of spaces remaining for the figure as drawn could be anywhere from 9 to 12. The value of 10-byte spaces has been chosen just for purposes of illustration.) After the first linear burst transfer, 6 byte spaces remain. Knowing that the second transfer will use another 4 bytes of space, the PCnet-PCI controller is able to predict that the third transfer will be the last. Therefore, it de-asserts FRAME on the second transfer to terminate the linear burst operation. However, the PCnet-PCI controller retains ownership of the bus so that it may, immediately, make non-burst transfer(s) to the last two spaces in the buffer.



o DEVSEL *is sampled by the PCnet-PCI controller.*

18220C-19

Figure 17. Partial Linear Burst at the End of Buffer

If a burst cycle is preempted before the last data phase, the PCnet-PCI controller will finish the current data phase and release the bus. REQ will stay asserted. The PCnet-PCI controller will revert to non-burst cycle accesses following the preemption event. In this case, linear bursting will next occur when the memory address being accessed next meets the linear burst starting address requirements.

Slave I/O Transfers

After the PCnet-PCI controller is configured as I/O device (by setting IOEN in the PCI Command register), it starts monitoring the PCI bus for access to its internal registers. The PCnet-PCI controller will look for an address that falls within its 32 bytes of I/O address space (starting from the I/O base address). The PCnet-PCI controller will assert $\overline{\text{DEVSEL}}$ if it detects an address match and the access is an I/O cycle. $\overline{\text{DEVSEL}}$ is asserted two clock cycles after the host has asserted $\overline{\text{FRAME}}$. The PCnet-PCI controller will not assert $\overline{\text{DEVSEL}}$ if it detects an address match, but the PCI command is not of the type I/O read or I/O write. The PCnet-PCI controller will suspend looking for I/O cycles while being a bus master.

Slave I/O Read

The Slave I/O Read command is used by the host CPU to read the PCnet-PCI's CSRs, BCRs and EEPROM locations. It is a single cycle, non-burst 8-bit,16-bit or 32-bit transfer which is initiated by the host CPU. The typical number of wait states added to a slave I/O read access on the part of the PCnet-PCI controller is 6 to 7 clock cycles, depending upon the relative phases of the internal Buffer Management Unit clock and the CLK signal, since the internal Buffer Management Unit clock is a divide-by-two version of the CLK signal. The PCnet-PCI controller will not produce Slave I/O Read commands while being a bus master.

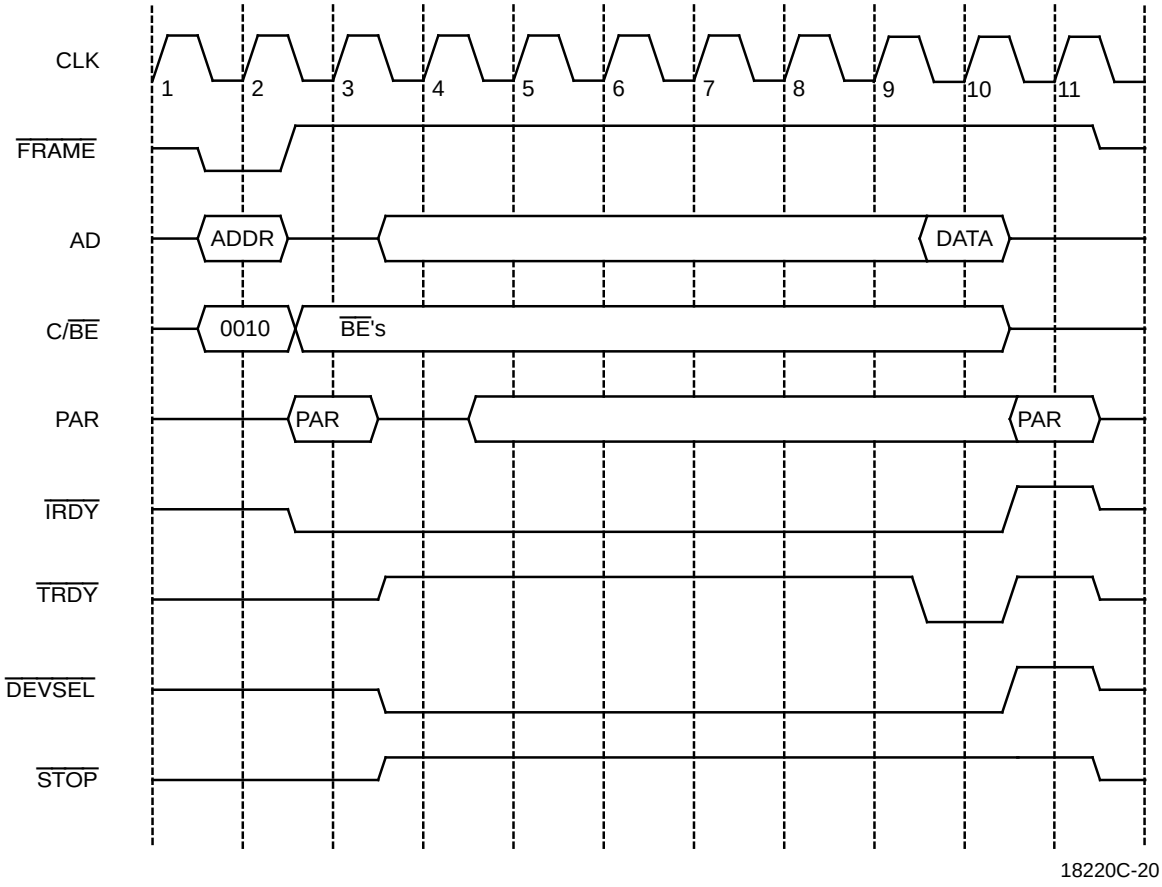


Figure 18. Slave I/O Read

Slave I/O Write

The Slave I/O Write command is used by the host CPU to write to the PCnet-PCI's CSRs, BCRs and EEPROM locations. It is a single cycle, non-burst 16-bit or 32-bit transfer which is initiated by the host CPU. The typical number of wait states added to a slave I/O write access on the part of the PCnet-PCI controller is 6 to 7 clock cycles, depending upon the relative phases of the internal Buffer Management Unit clock and the CLK signal, since the internal Buffer Management Unit clock is a divide-by-two version of the CLK signal. The PCnet-PCI controller will not produce Slave I/O write commands while being a bus master.

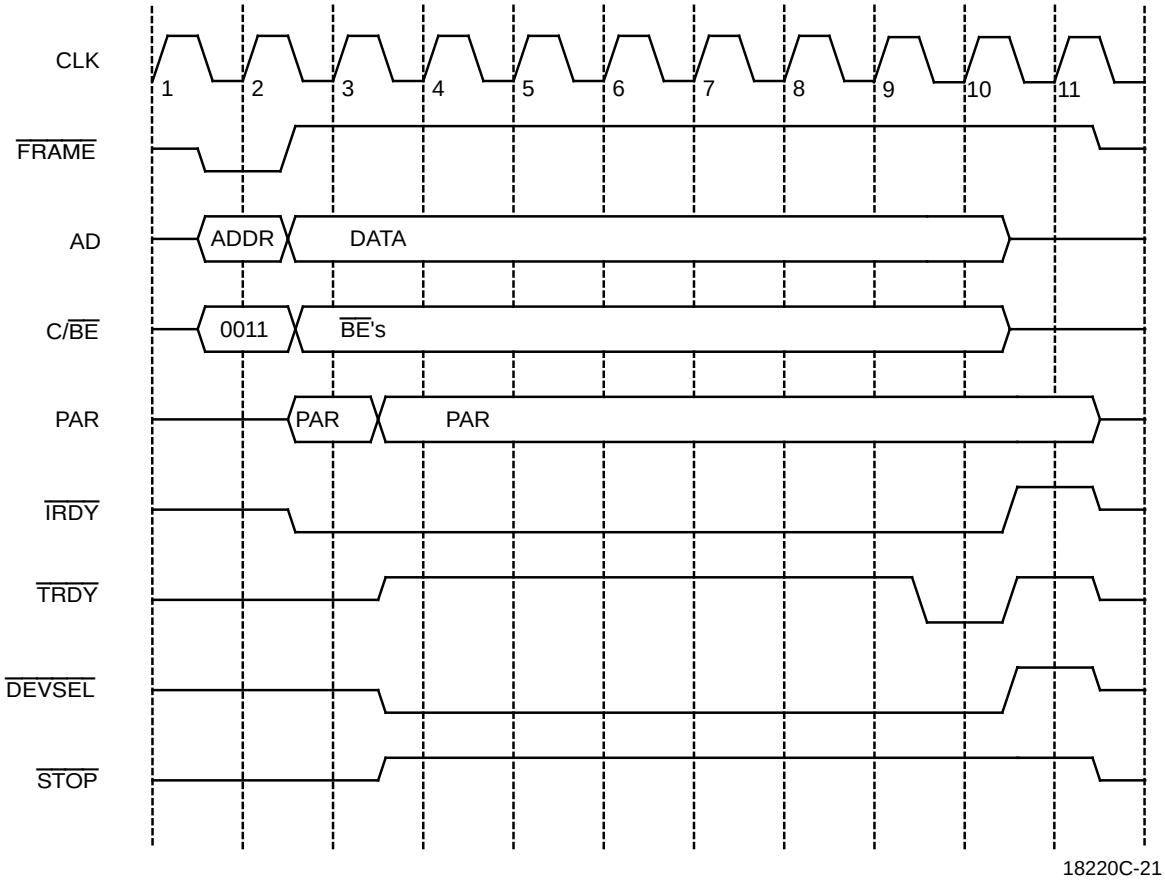


Figure 19. Slave I/O Write

Slave Configuration Transfers

The host can access the PCnet-PCI PCI configuration space with a configuration read or write command. The PCnet-PCI controller will assert DEVSEL if the IDSEL input is asserted during the address phase and if the access is a configuration cycle. DEVSEL is asserted two clock cycles after the host has asserted FRAME. All configuration cycles are of fixed length. The PCnet-PCI controller will assert TRDY on the 3rd clock of the data phase.

Slave Configuration Read

The Slave Configuration Read command is used by the host CPU to read the configuration space in the PCnet-PCI controller. This provides the host CPU with information concerning the device and its capabilities. This is a single cycle, non-burst 8-bit, 16-bit, or 32-bit transfer.

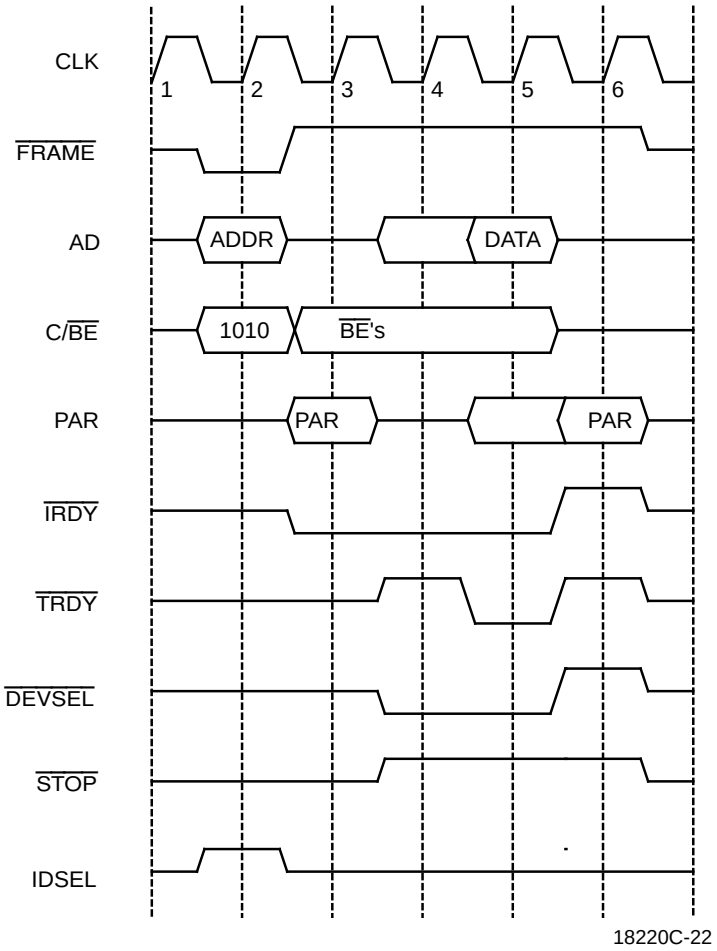


Figure 20. Slave Configuration Read

Slave Configuration Write

The Slave Configuration Write command is used by the host CPU to write the configuration space in the PCnet-PCI controller. This allows the host CPU to control basic activity of the device, such as enable/disable, change I/O location, etc. This is a single cycle, non-burst 8-bit, 16-bit, or 32-bit transfer.

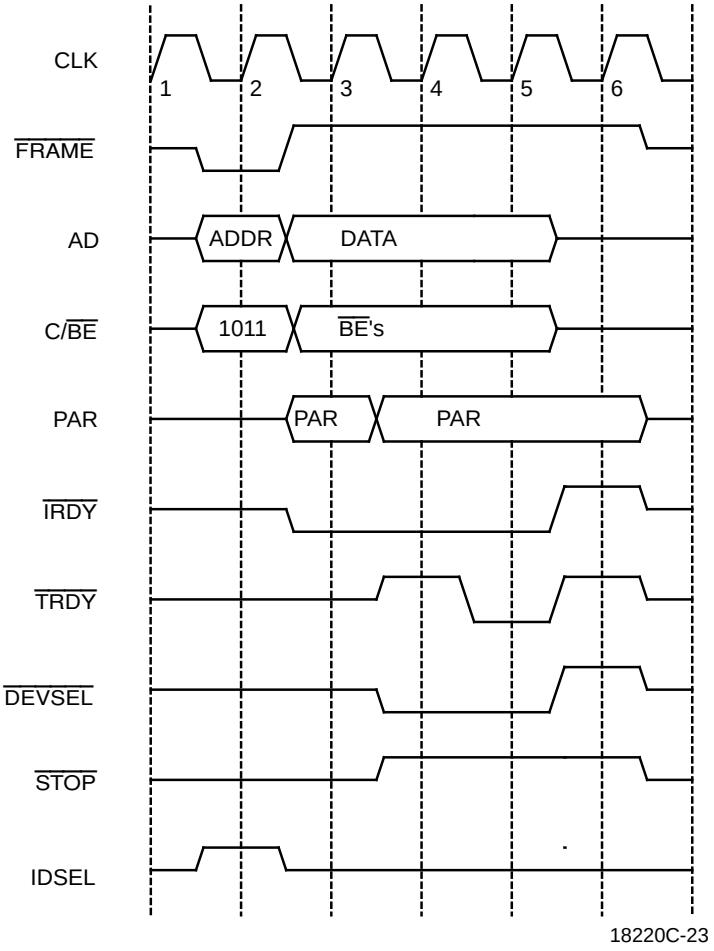


Figure 21. Slave Configuration Write

Buffer Management Unit (BMU)

The buffer management unit is a micro-coded state machine which implements the initialization procedure and manages the descriptors and buffers. The buffer management unit operates at half the speed of the CLK input.

Initialization

PCnet-PCI initialization includes the reading of the initialization block in memory to obtain the operating parameters. The initialization block is read when the INIT bit in CSR0 is set. The INIT bit should be set before or concurrent with the STRT bit to insure correct operation. Two DWORDs are read during each period of bus mastership. When SSIZE32 = 1 (BCR20, bit 8), this results in a total of 4 arbitration cycles (3 arbitration cycles if SSIZE32 = 0). Once the initialization block has been completely read in and internal registers have been updated, IDON will be set in CSR0, and an interrupt generated (if IENA is set). At this point, the BMU knows where the receive and transmit descriptor rings and hence, normal network operations will begin.

The PCnet-PCI controller obtains the start address of the Initialization Block from the contents of CSR1 (least significant 16 bits of address) and CSR2 (most significant 16 bits of address). The host must write CSR1 and CSR2 before setting the INIT bit. The block contains the user defined conditions for PCnet-PCI operation, together with the base addresses and length information of the transmit and receive descriptor rings.

There is an alternative method to initialize the PCnet-PCI controller. Instead of initialization via the initialization block in memory, data can be written directly into the appropriate registers. Either method may be used at the discretion of the programmer. If the registers are written to directly, the INIT bit must not be set, or the initialization block will be read in, thus overwriting the previously written information. Please refer to Appendix C for details on this alternative method.

If initialization is done by writing directly to registers, the Polling Interval register (CSR47) must be initialized in addition to those registers that can be loaded automatically from the initialization block.

Re-Initialization

The transmitter and receiver sections of the PCnet-PCI controller can be turned on via the initialization block (MODE Register DTX, DRX bits; CSR15[1:0]). The states of the transmitter and receiver are monitored by the host through CSR0 (RXON, TXON bits). The PCnet-PCI controller should be reinitialized if the transmitter and/or the receiver were not turned on during the original initialization, and it was subsequently required to activate them or if either section was shut off due to the detection of an error condition (MERR, UFLO, TX BUFF error).

Reinitialization may be done via the initialization block or by setting the STOP bit in CSR0, followed by writing to CSR15, and then setting the START bit in CSR0. Note that this form of restart will not perform the same in the PCnet-PCI controller as in the LANCE. In particular, upon restart, the PCnet-PCI controller reloads the transmit and receive descriptor pointers with their respective base addresses. This means that the software must clear the descriptor own bits and reset its descriptor ring pointers before the restart of the PCnet-PCI controller. The reload of descriptor base addresses is performed in the LANCE only after initialization, so a restart of the LANCE without initialization leaves the LANCE pointing at the same descriptor locations as before the restart.

Buffer Management

Buffer management is accomplished through message descriptor entries organized as ring structures in memory. There are two rings, a receive ring and a transmit ring. The size of a message descriptor entry is 4 DWORDs, or 16 bytes, when SSIZE32 = 1. The size of a message descriptor entry is 4 words, or 8 bytes, when SSIZE32 = 0.

Descriptor Rings

Each descriptor ring must be organized in a contiguous area of memory. At initialization time (setting the INIT bit in CSR0), the PCnet-PCI controller reads the user-defined base address for the transmit and receive descriptor rings, as well as the number of entries contained in the descriptor rings. Descriptor ring base addresses must be on a 16-byte boundary when SSIZE32=1, and 8-byte boundary when SSIZE=0. A maximum of 128 (or 512, depending upon the value of SSIZE32) ring entries is allowed when the ring length is set through the TLEN and RLEN fields of the initialization block. However, the ring lengths can be set beyond this range (up to 65535) by writing the transmit and receive ring length registers (CSR76, CSR78) directly.

Each ring entry contains the following information:

1. The address of the actual message data buffer in user or host memory
2. The length of the message buffer
3. Status information indicating the condition of the buffer

To permit the queuing and de-queuing of message buffers, ownership of each buffer is allocated to either the PCnet-PCI controller or the host. The OWN bit within the descriptor status information, either TMD or RMD (see section on TMD or RMD), is used for this purpose. OWN = “1” signifies that the PCnet-PCI controller currently has ownership of this ring descriptor and its associated buffer. Only the owner is permitted to relinquish ownership or to write to any field in the descriptor entry. A device that is not the current owner of a descriptor entry cannot assume ownership or change any field in the

entry. A device may, however, read from a descriptor that it does not currently own. Software should always read descriptor entries in sequential order. When software finds that the current descriptor is owned by the PCnet-PCI controller, then the software must not read "ahead" to the next descriptor. The software should wait at the unOWNed descriptor until ownership has been granted to the software (when SPRINTEN = 1 (CSR3, bit 5), then this rule is modified. See the SPRINTEN description). Strict adherence to these rules insures that "Deadly Embrace" conditions are avoided.

Descriptor Ring Access Mechanism

At initialization, the PCnet-PCI controller reads the base address of both the transmit and receive descriptor rings into CSRs for use by the PCnet-PCI controller during subsequent operations.

As the final step in the self-initialization process, the base address of each ring is loaded into each of the current descriptor address registers and the address of the next descriptor entry in the transmit and receive rings is computed and loaded into each of the next descriptor address registers.

When SSIZE32 = 0, software data structures are 16 bits wide. The following diagram, Figure 22, illustrates the relationship between the Initialization Base Address, the Initialization Block, the Receive and Transmit Descriptor Ring Base Addresses, the Receive and Transmit Descriptors and the Receive and Transmit Data Buffers, for the case of SSIZE32 = 0.

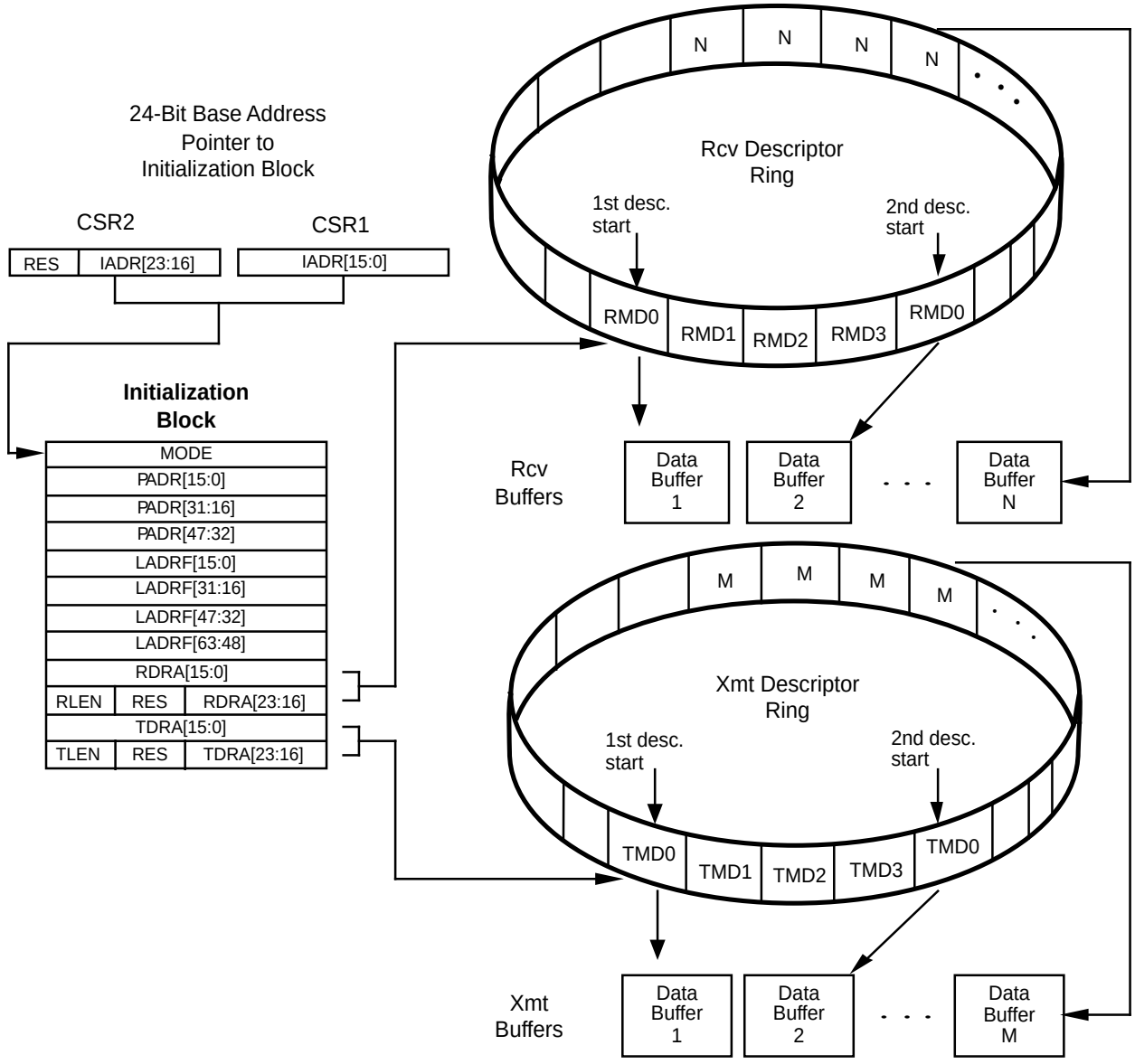


Figure 22. 16-Bit Data Structures: Initialization Block and Descriptor Rings 18220C-24

When SSIZE32 = 1, software data structures are 32 bits wide. The following diagram illustrates, Figure 23, the relationship between the Initialization Base Address, the Initialization Block, the Receive and Transmit Descriptor Ring Base Addresses, the Receive and Transmit Descriptors and the Receive and Transmit Data Buffers, for the case of SSIZE32 = 1.

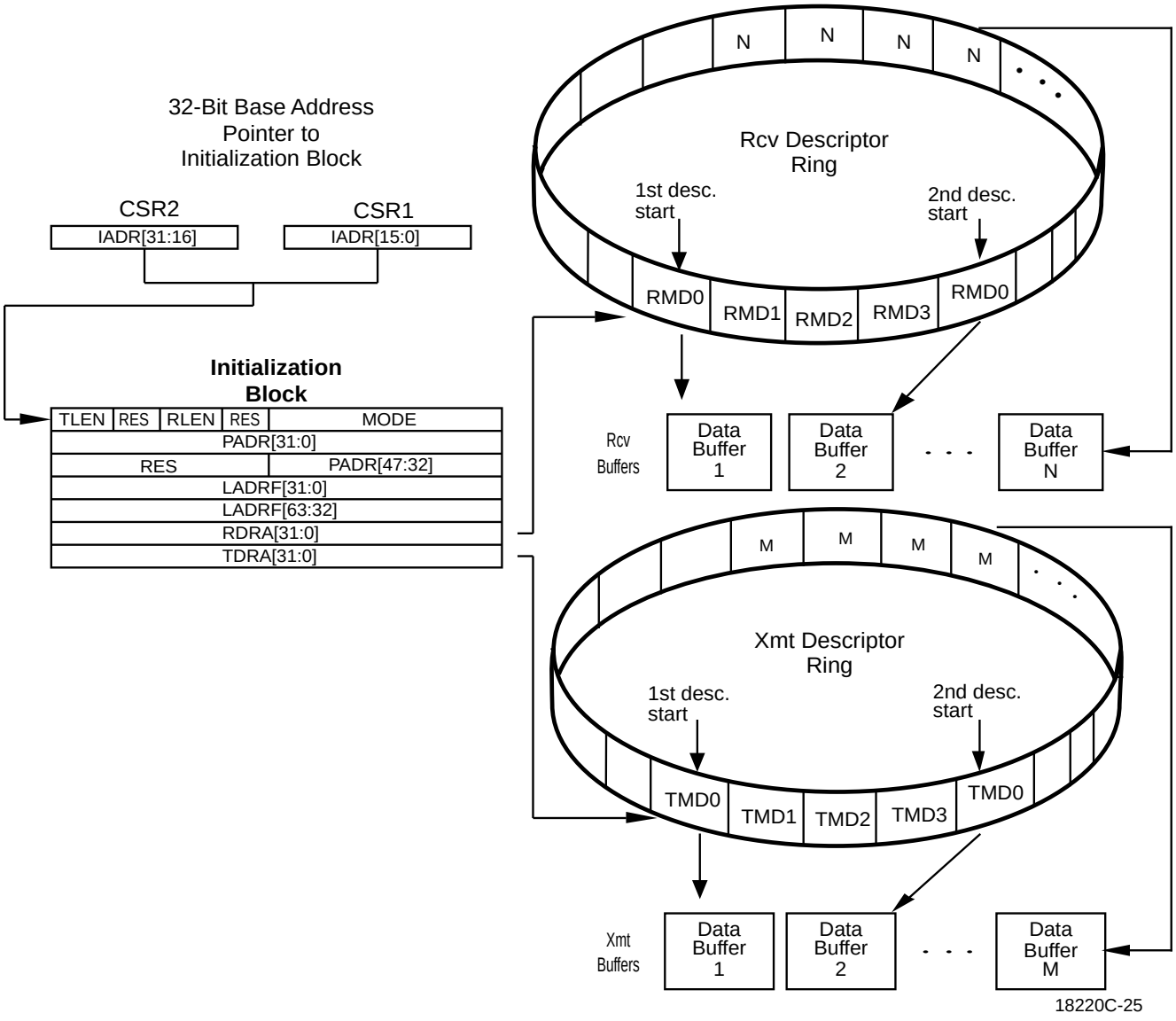


Figure 23. 32-Bit Data Structures: Initialization Block and Descriptor Rings

Polling

If there is no network channel activity and there is no pre- or post-receive or pre- or post-transmit activity being performed by the PCnet-PCI controller, then the PCnet-PCI controller will periodically poll the current receive and transmit descriptor entries in order to ascertain their ownership. If the DPOLL bit in CSR4 is set, then the transmit polling function is disabled.

A typical polling operation consists of the following: The PCnet-PCI controller will use the current receive descriptor address stored internally to vector to the appropriate Receive Descriptor Table Entry (RDTE). It will then use the current transmit descriptor address (stored internally) to vector to the appropriate Transmit Descriptor Table Entry (TDTE). The accesses will be made in the following order: RMD1, then RMD0 of the current RDTE during one bus arbitration, and after that, TMD1, then TMD0 of the current TDTE during a second bus arbitration. All information collected during polling activity will be stored internally in the appropriate CSRs. (i.e. CSR18, CSR19, CSR20, CSR21, CSR40, CSR42, CSR50, CSR52). UnOWNed descriptor status will be internally ignored.

A typical receive poll is the product of the following conditions:

1. PCnet-PCI controller does not possess ownership of the current RDTE and the poll time has elapsed and RXON=1 (CSR0, bit 5), or
2. PCnet-PCI controller does not possess ownership of the next RDTE the poll time has elapsed and RXON=1.

If RXON=0 the PCnet-PCI controller will never poll RDTE locations.

The ideal system should always have at least one RDTE available for the possibility of an unpredictable receive event. (This condition is not a requirement. If this condition is not met, it simply means that frames will be missed by the system because there was no buffer space available.) But the typical system usually has at least one or two RDTEs available for the possibility of an unpredictable receive event. Given that this condition is satisfied, the current and next RDTE polls are rarely seen and hence, the typical poll operation simply consists of a check of the status of the current TDTE. When there is only one RDTE (because the RLEN was set to ZERO), then there is no “next RDTE” and ownership of “next RDTE” cannot be checked. If there is at least one RDTE, the RDTE poll will rarely be seen and the typical poll operation simply consists of a check of the current TDTE.

A typical transmit poll is the product of the following conditions:

1. PCnet-PCI controller does not possess ownership of the current TDTE and
   DPOLL=0 (CSR4, bit 2) and
   TXON=1 (CSR0, bit 4) and
   the poll time has elapsed, or
2. PCnet-PCI controller does not possess ownership of the current TDTE and
   DPOLL=0 and
   TXON=1 and
   a frame has just been received, or
3. PCnet-PCI controller does not possess ownership of the current TDTE and
   DPOLL=0 and
   TXON=1 and
   a frame has just been transmitted.

Setting the TDMD bit of CSR0 will cause the microcode controller to exit the poll counting code and immediately perform a polling operation. If RDTE ownership has not been previously established, then an RDTE poll will be performed ahead of the TDTE poll. If the microcode is not executing the poll counting code when the TDMD bit is set, then the demanded poll of the TDTE will be delayed until the microcode returns to the poll counting code.

The user may change the poll time value from the default of 65,536 clock periods by modifying the value in the Polling Interval register (CSR47). Note that if a non–default value is desired, then a strict sequence of setting the INIT bit in CSR0, waiting for IDONE, then writing to CSR47, and then setting STRT in CSR0 must be observed, otherwise the default value will not be overwritten. See the CSR47 section for details.

Transmit Descriptor Table Entry (TDTE)

If, after a TDTE access, the PCnet-PCI controller finds that the OWN bit of that TDTE is not set, then the PCnet-PCI controller resumes the poll time count and reexamines the same TDTE at the next expiration of the poll time count.

If the OWN bit of the TDTE is set, but Start of Frame (STP) bit is not set, the PCnet-PCI controller will immediately request the bus in order to reset the OWN bit of this descriptor. (This condition would normally be found following a LCOL or RETRY error that occurred in the middle of a transmit frame chain of buffers.) After resetting the OWN bit of this descriptor, the PCnet-PCI controller will again immediately request the bus in order to access the next TDTE location in the ring.

If the OWN bit is set and the buffer length is 0, the OWN bit will be reset. In the LANCE the buffer length of 0 is interpreted as a 4096-byte buffer. It is acceptable to have a 0 length buffer on transmit with STP = 1 or STP = 1 *and* ENP = 1. It is not acceptable to have 0 length buffer with STP = 0 and ENP =1.

If the OWN bit is set and the start of frame (STP) bit is set, then microcode control proceeds to a routine that will enable transmit data transfers to the FIFO. The PCnet-PCI controller will look ahead to the next transmit descriptor after it has performed at least one transmit data transfer from the first buffer. (More than one transmit data transfer may possibly take place, depending upon the state of the transmitter.) The contents of TMD0 and TMD1 will be stored in Next Xmt Buffer Address (CSR64 and CSR65), Next Xmt Byte Count (CSR66) and Next Xmt Status (CSR67) regardless of the state of the OWN bit. This transmit descriptor lookahead operation is performed only once.

If the PCnet-PCI controller does not own the next TDTE (i.e. the second TDTE for this frame), then it will complete transmission of the current buffer and then update the status of the current (first) TDTE with the BUFF and UFLO bits being set. This will cause the transmitter to be disabled (CSR0, TXON=0). The PCnet-PCI controller will have to be re-initialized to restore the transmit function. The situation that matches this description implies that the system has not been able to stay ahead of the PCnet-PCI controller in the transmit descriptor ring and therefore, the condition is treated as a fatal error. (To avoid this situation, the system should always set the transmit chain descriptor own bits in reverse order.)

If the PCnet-PCI controller does own the second TDTE in a chain, it will gradually empty the contents of the first buffer (as the bytes are needed by the transmit operation), perform a single-cycle DMA transfer to update the status of the first descriptor (reset the OWN bit in TMD1), and then it may perform one data DMA access on the second buffer in the chain before executing another lookahead operation. (i.e. a lookahead to the third descriptor.)

The PCnet-PCI controller can queue up to two frames in the transmit FIFO. Call them frame “X” and frame “Y”, where “Y” is after “X”. Assume that frame “X” is currently being transmitted. Because the PCnet-PCI controller can perform lookahead data transfer past the ENP of frame “X”, it is possible for the PCnet-PCI controller to completely transfer the data from a buffer belonging to frame “Y” into the FIFO even though frame “X” has not yet been completely transmitted. At the end of this “Y” buffer data transfer, the PCnet-PCI controller will write intermediate status (change the OWN bit to a ZERO) for the “Y” frame buffer, if frame “Y” uses data chaining. The last TDTE for the “X” frame (containing ENP) has not yet been written, since the “X” frame has not yet been completely transmitted. Note that the PCnet-PCI controller has, in this instance, returned ownership of a TDTE to the host out of a “normal” sequence.

For this reason, it becomes imperative that the host system should never read the Transmit DTE ownership bits out of order. Software should always process buffers in sequence, waiting for the ownership before proceeding.

There should be no problems for software which processes buffers in sequence, waiting for ownership before proceeding.

If an error occurs in the transmission before all of the bytes of the current buffer have been transferred, then TMD2 and TMD1 of the current buffer will be written; In such a case, data transfers from the next buffer will not commence. Instead, following the TMD2/TMD1 update, the PCnet-PCI controller will go to the next transmit frame, if any, skipping over the rest of the frame which experienced an error, including chained buffers. This is done by returning to the polling microcode where PCnet-PCI controller will immediately access the next descriptor and find the condition OWN=1 and STP=0 as described earlier. As described for that case, the PCnet-PCI controller will reset the own bit for this descriptor and continue in like manner until a descriptor with OWN=0 (no more transmit frames in the ring) or OWN=1 and STP=1 (the first buffer of a new frame) is reached.

At the end of any transmit operation, whether successful or with errors, immediately following the completion of the descriptor updates, the PCnet-PCI controller will always perform another poll operation. As described earlier, this poll operation will begin with a check of the current RDTE, unless the PCnet-PCI controller already owns that descriptor. Then the PCnet-PCI controller will proceed to polling the next TDTE. If the transmit descriptor OWN bit has a ZERO value, then the PCnet-PCI controller will resume poll time count incrementing. If the transmit descriptor OWN bit has a value of ONE, then the PCnet-PCI controller will begin filling the FIFO with transmit data and initiate a transmission. This end–of–operation poll coupled with the TDTE lookahead operation allows the PCnet-PCI controller to avoid inserting poll time counts between successive transmit frames.

Whenever the PCnet-PCI controller completes a transmit frame (either with or without error) and writes the status information to the current descriptor, then the TINT bit of CSR0 is set to indicate the completion of a transmission. This causes an interrupt signal if the IENA bit of CSR0 has been set and the TINTM bit of CSR3 is reset.

Receive Descriptor Table Entry (RDTE)

If the PCnet-PCI controller does not own both the current and the next Receive Descriptor Table Entry then the PCnet-PCI controller will continue to poll according to the polling sequence described above. If the receive descriptor ring length is 1, then there is no next descriptor to be polled.

If a poll operation has revealed that the current and the next RDTE belong to the PCnet-PCI controller then additional poll accesses are not necessary. Future poll operations will not include RDTE accesses as long as the PCnet-PCI controller retains ownership of the current and the next RDTE.

When receive activity is present on the channel, the PCnet-PCI controller waits for the complete address of the message to arrive. It then decides whether to accept or reject the frame based on all active addressing schemes. If the frame is accepted the PCnet-PCI controller checks the current receive buffer status register CRST (CSR41) to determine the ownership of the current buffer.

If ownership is lacking, then the PCnet-PCI controller will immediately perform a (last ditch) poll of the current RDTE. If ownership is still denied, then the PCnet-PCI controller has no buffer in which to store the incoming message. The MISS bit will be set in CSR0 and an interrupt will be generated if IENA=1 (CSR0) and MISSM=0 (CSR3). Another poll of the current RDTE will not occur until the frame has finished.

If the PCnet-PCI controller sees that the last poll (either a normal poll, or the last-ditch effort described in the above paragraph) of the current RDTE shows valid ownership, then it proceeds to a poll of the next RDTE. Following this poll, and regardless of the outcome of this poll, transfers of receive data from the FIFO may begin.

Regardless of ownership of the second receive descriptor, the PCnet-PCI controller will continue to perform receive data DMA transfers to the first buffer. If the frame length exceeds the length of the first buffer, and the PCnet-PCI controller does not own the second buffer, ownership of the current descriptor will be passed back to the system by writing a ZERO to the OWN bit of RMD1 and status will be written indicating buffer (BUFF=1) and possibly overflow (OFLO=1) errors.

If the frame length exceeds the length of the first (current) buffer, and the PCnet-PCI controller does own the second (next) buffer, ownership will be passed back to the system by writing a ZERO to the OWN bit of RMD1 when the first buffer is full. Receive data transfers to the second buffer may occur before the PCnet-PCI controller proceeds to look ahead to the ownership of the third buffer. Such action will depend upon the state of the FIFO when the status has been updated on the first descriptor. In any case, lookahead will be performed to the third buffer and the information gathered will be stored in the chip, regardless of the state of the ownership bit. As in the transmit flow, lookahead operations are performed only once.

This activity continues until the PCnet-PCI controller recognizes the completion of the frame (the last byte of this receive message has been removed from the FIFO). The PCnet-PCI controller will subsequently update the current RDTE status with the end of frame (ENP) indication set, write the message byte count (MCNT) of the complete frame into RMD2 and overwrite the “current” entries in the CSRs with the “next” entries.

Media Access Control

The Media Access Control engine incorporates the essential protocol requirements for operation of a compliant Ethernet/802.3 node, and provides the interface between the FIFO sub-system and the Manchester Encoder/Decoder (MENDEC).

The MAC engine is fully compliant to Section 4 of ISO/IEC 8802-3 (ANSI/IEEE Standard 1990 Second edition) and ANSI/IEEE 802.3 (1985).

The MAC engine provides programmable enhanced features designed to minimize host supervision, bus utilization, and pre- or post- message processing. These include the ability to disable retries after a collision, dynamic FCS generation on a frame-by-frame basis, and automatic pad field insertion and deletion to enforce minimum frame size attributes, automatic retransmission without reloading the FIFO, automatic deletion of collision fragments, and reduces bus bandwidth use.

The two primary attributes of the MAC engine are:

- Transmit and receive message data encapsulation.
  - — Framing (frame boundary delimitation, frame synchronization).
  - — Addressing (source and destination address handling).
  - — Error detection (physical medium transmission errors).
- Media access management.
  - — Medium allocation (collision avoidance).
  - — Contention resolution (collision handling).

Transmit and Receive Message Data Encapsulation

The MAC engine provides minimum frame size enforcement for transmit and receive frames. When APAD_XMT = 1 (CSR, bit 11), transmit messages will be padded with sufficient bytes (containing 00h) to ensure that the receiving station will observe an information field (destination address, source address, length/type, data and FCS) of 64-bytes. When ASTRP_RCV = 1 (CSR4, bit 10), the receiver will

automatically strip pad bytes from the received message by observing the value in the length field, and stripping excess bytes if this value is below the minimum data size (46 bytes). Both features can be independently over-ridden to allow illegally short (less than 64 bytes of frame data) messages to be transmitted and/or received. The use of this feature reduces bus utilization because the pad bytes are not transferred into or out of main memory.

Framing (Frame Boundary Delimitation, Frame Synchronization)

The MAC engine will autonomously handle the construction of the transmit frame. Once the Transmit FIFO has been filled to the predetermined threshold (set by XMTSP in CSR80), and providing access to the channel is currently permitted, the MAC engine will commence the 7 byte preamble sequence (10101010b, where first bit transmitted is a 1). The MAC engine will subsequently append the Start Frame Delimiter (SFD) byte (10101011b) followed by the serialized data from the Transmit FIFO. Once the data has been completed, the MAC engine will append the FCS (most significant bit first) which was computed on the entire data portion of the frame. The data portion of the frame consists of destination address, source address, length/type, and frame data.

The user is responsible for the correct ordering and content in each of the fields in the frame.

The receive section of the MAC engine will detect an incoming preamble sequence and lock to the encoded clock. The internal MENDEC will decode the serial bit stream and present this to the MAC engine. The MAC will discard the first 8-bits of information before searching for the SFD sequence. Once the SFD is detected, all subsequent bits are treated as part of the frame. The MAC engine will inspect the length field to ensure minimum frame size, strip unnecessary pad characters (if enabled), and pass the remaining bytes through the Receive FIFO to the host. If pad stripping is performed, the MAC engine will also strip the received FCS bytes, although the normal FCS computation and checking will occur. Note that apart from pad stripping, the frame will be passed unmodified to the host. If the length field has a value of 46 or greater, the MAC engine will not attempt to validate the length against the number of bytes contained in the message.

If the frame terminates or suffers a collision before 64-bytes of information (after SFD) have been received, the MAC engine will automatically delete the frame from the Receive FIFO, without host intervention. The PCnet-PCI controller has the ability to accept runt packets for diagnostics purposes and proprietary networks.

Addressing (Source and Destination Address Handling)

The first 6-bytes of information after SFD will be interpreted as the destination address field. The MAC engine provides facilities for physical, logical (multicast) and broadcast address reception.

Error Detection (Physical Medium Transmission Errors)

The MAC engine provides several facilities which report and recover from errors on the medium. In addition, the network is protected from gross errors due to inability of the host to keep pace with the MAC engine activity.

On completion of transmission, the following transmit status is available in the appropriate TMD and CSR areas:

- The exact number of transmission retry attempts (ONE, MORE, RTRY or TRC).
- Whether the MAC engine had to Defer (DEF) due to channel activity.
- Excessive deferral (EXDEF), indicating that the transmitter has experienced Excessive Deferral on this transmit frame, where Excessive Deferral is defined in ISO 8802-3 (IEEE/ANSI 802.3).
- Loss of Carrier (LCAR), indicating that there was an interruption in the ability of the MAC engine to monitor its own transmission. Repeated LCAR errors indicate a potentially faulty transceiver or network connection.
- Late Collision (LCOL) indicates that the transmission suffered a collision after the slot time. This is indicative of a badly configured network. Late collisions should not occur in a normal operating network.
- Collision Error (CERR) indicates that the transceiver did not respond with an SQE Test message within the predetermined time after a transmission completed. This may be due to a failed transceiver, disconnected or faulty transceiver drop cable, or the fact the transceiver does not support this feature (or it is disabled).

In addition to the reporting of network errors, the MAC engine will also attempt to prevent the creation of any network error due to the inability of the host to service the MAC engine. During transmission, if the host fails to keep the Transmit FIFO filled sufficiently, causing an underflow, the MAC engine will guarantee the message is either sent as a runt packet (which will be deleted by the receiving station) or has an invalid FCS (which will also cause the receiver to reject the message).

The status of each receive message is available in the appropriate RMD and CSR areas. FCS and Framing errors (FRAM) are reported, although the received frame is still passed to the host. The FRAM error will only be reported if an FCS error is detected and there are a non integral number of bytes in the message. The MAC engine will ignore up to 7 additional bits at the end of a message (dribbling bits), which can occur under normal network operating conditions. The reception of 8 additional bits will cause the MAC engine to de-serialize the entire byte, and will result in the received message and FCS being modified.

The PCnet-PCI controller can handle up to 7 dribbling bits when a received frame terminates. During the reception, the FCS is generated on every serial bit (including the dribbling bits) coming from the cable, although the internally saved FCS value is only updated on the eighth bit (on each byte boundary). The framing error is reported to the user as follows:

- If the number of dribbling bits are 1 to 7 and there is no CRC (FCS) error, then there is no Framing error (FRAM = 0).
- If the number of dribbling bits are 1 to 7 and there is a CRC (FCS) error, then there is also a Framing error (FRAM = 1).
- If the number of dribbling bits = 0, then there is no Framing error. There may or may not be a CRC (FCS) error.

Counters are provided to report the Receive Collision Count and Runt Packet Count for network statistics and utilization calculations.

Note that if the MAC engine detects a received frame which has a 00b pattern in the preamble (after the first 8 bits which are ignored), the entire frame will be ignored. The MAC engine will wait for the network to go inactive before attempting to receive additional frames.

Media Access Management

The basic requirement for all stations on the network is to provide fairness of channel allocation. The 802.3/Ethernet protocols define a media access mechanism which permits all stations to access the channel with equality. Any node can attempt to contend for the channel by waiting for a predetermined time (Inter Packet Gap internal) after the last activity, before transmitting on the media. The channel is a multidrop communications media (with various topological configurations permitted) which allows a single station to transmit and all other stations to receive. If two nodes simultaneously contend for the channel, their signals will interact causing loss of data, defined as a collision. It is the responsibility of the MAC to attempt to avoid and recover from a collision, to guarantee data integrity for the end-to-end transmission to the receiving station.

Medium Allocation

The IEEE/ANSI 802.3 Standard (ISO/IEC 8802-3 1990) requires that the CSMA/CD MAC monitor the medium for traffic by watching for carrier activity. When carrier is detected, the media is considered busy, and the MAC should defer to the existing message.

The ISO 8802-3 (IEEE/ANSI 802.3) Standard also allows optional two part deferral after a receive message.

See ANSI/IEEE Std 802.3 –1990 Edition, 4.2.3.2.1:

***Note**: It is possible for the PLS carrier sense indication to fail to be asserted during a collision on the media. If the deference process simply times the interFrame gap based on this indication it is possible for a short interFrame gap to be generated, leading to a potential reception failure of a subsequent frame. To enhance system robustness the following optional measures, as specified in 4.2.8, are recommended when InterFrameSpacingPart1 is other than ZERO:*

1. *Upon completing a transmission, start timing the interpacket gap, as soon as transmitting and carrier Sense are both false.*
2. *When timing an interFrame gap following reception, reset the interFrame gap timing if carrier Sense becomes true during the first 2/3 of the interFrame gap timing interval. During the final 1/3 of the interval the timer shall not be reset to ensure fair access to the medium. An initial period shorter than 2/3 of the interval is permissible including ZERO.*

The MAC engine implements the optional receive two part deferral algorithm, with a first part inter-frame-spacing time of 6.0 µs. The second part of the inter-frame-spacing interval is therefore 3.6 µs.

The PCnet-PCI controller will perform the two part deferral algorithm as specified in Section 4.2.8 (Process Deference). The Inter Packet Gap (IPG) timer will start timing the 9.6 µs InterFrameSpacing after the receive carrier is de-asserted. During the first part deferral (InterFrameSpacingPart1 – IFS1) the PCnet-PCI controller will defer any pending transmit frame and respond to the receive message. The IPG counter will be reset to ZERO continuously until the carrier de-asserts, at which point the IPG counter will resume the 9.6 µs count once again. Once the IFS1 period of 6.0 µs has elapsed, the PCnet-PCI controller will begin timing the second part deferral (InterFrame Spacing Part 2 – IFS2) of 3.6 µs. Once IFS1 has completed, and IFS2 has commenced, the PCnet-PCI controller will not defer to a receive frame if a transmit frame is pending. This means that the PCnet-PCI controller will not attempt to receive the receive frame, since it will start to transmit, and generate a collision at 9.6 µs. The PCnet-PCI controller will guarantee to complete the preamble (64-bit) and jam (32-bit)

sequence before ceasing transmission and invoking the random backoff algorithm.

This transmit two part deferral algorithm is implemented as an option which can be disabled using the DXMT2PD bit in CSR3. Two part deferral after transmission is useful for ensuring that severe IPG shrinkage cannot occur in specific circumstances, causing a transmit message to follow a receive message so closely as to make them indistinguishable.

During the time period immediately after a transmission has been completed, the external transceiver (in the case of a standard AUI connected device), should generate the SQE Test message (a nominal 10 MHz burst of 5 –15 Bit Times duration) on the CI± pair (within 0.6 µs – 1.6 µs after the transmission ceases). During the time period in which the SQE Test message is expected the PCnet-PCI controller will not respond to receive carrier sense.

See ANSI/IEEE Std 802.3—1990 Edition, 7.2.4.6 (1):

"At the conclusion of the output function, the DTE opens a time window during which it expects to see the signal_quality_error *signal asserted on the Control In circuit. The time window begins when the CARRIER_STATUS becomes CARRIER_OFF. If execution of the output function does not cause CARRIER_ON to occur, no SQE test occurs in the DTE. The duration of the window shall be at least 4.0 µs but no more than 8.0 µs. During the time window the Carrier Sense Function is inhibited."*

The PCnet-PCI controller implements a carrier sense "blinding" period within 0 µs – 4.0 µs from de-assertion of carrier sense after transmission. This effectively means that when transmit two part deferral is enabled (DXMT2PD is cleared) the IFS1 time is from 4 µs to 6 µs after a transmission. However, since IPG shrinkage below 4 µs will rarely be encountered on a correctly configured networks, and since the fragment size will be larger than the 4 µs blinding window, then the IPG counter will be reset by a worst case IPG shrinkage/fragment scenario and the PCnet-PCI controller will defer its transmission. In addition, the PCnet-PCI controller will not restart the "blinding" period if carrier is detected within the 4.0 µs – 6.0 µs IFS1 period, but will commence timing of the entire IFS1 period.

Contention Resolution (Collision Handling)

Collision detection is performed and reported to the MAC engine by the integrated Manchester Encoder/Decoder (MENDEC). If a collision is detected before the complete preamble/SFD sequence has been transmitted, the MAC Engine will complete the preamble/SFD before appending the jam sequence. If a collision is detected after the preamble/SFD has been completed, but prior to 512 bits being transmitted, the MAC Engine will abort the transmission, and append the jam sequence immediately. The jam sequence is a 32-bit all Zeros pattern.

The MAC Engine will attempt to transmit a frame a total of 16 times (initial attempt plus 15 retries) due to normal collisions (those within the slot time). Detection of collision will cause the transmission to be re-scheduled, dependent on the backoff time that the MAC Engine computes. If a single retry was required, the ONE bit will be set in the Transmit Frame Status. If more than one retry was required, the MORE bit will be set. If all 16 attempts experienced collisions, the RTRY bit will be set (ONE and MORE will be clear), and the transmit message will be flushed from the FIFO. If retries have been disabled by setting the DRTY bit in CSR15, the MAC Engine will abandon transmission of the frame on detection of the first collision. In this case, only the RTRY bit will be set and the transmit message will be flushed from the FIFO.

If a collision is detected after 512 bit times have been transmitted, the collision is termed a late collision. The MAC Engine will abort the transmission, append the jam sequence and set the LCOL bit. No retry attempt will be scheduled on detection of a late collision, and the transmit message will be flushed from the FIFO.

The ISO 8802-3 (IEEE/ANSI 802.3) Standard requires use of a "truncated binary exponential backoff" algorithm which provides a controlled pseudo random mechanism to enforce the collision backoff interval, before re-transmission is attempted.

See ANSI/IEEE Std 802.3—1990 Edition, 4.2.3.2.5:

"At the end of enforcing a collision (jamming), the CSMA/CD sublayer delays before attempting to retransmit the frame. The delay is an integer multiple of slot Time. The number of slot times to delay before the nth re-transmission attempt is chosen as a uniformly distributed random integer r in the range:

$$0 \leq r < 2k$$

where

$$k = \min(n,10).$$"

The PCnet-PCI controller provides an alternative algorithm, which suspends the counting of the slot time/IPG during the time that receive carrier sense is detected. This aids in networks where large numbers of nodes are present, and numerous nodes can be in collision. It effectively accelerates the increase in the backoff time in busy networks, and allows nodes not involved in the collision to access the channel whilst the colliding nodes await a reduction in channel activity. Once channel activity is reduced, the nodes resolving the collision time out their slot time counters as normal.

Manchester Encoder/Decoder (MENDEC)

The integrated Manchester Encoder/Decoder provides the PLS (Physical Layer Signaling) functions required for a fully compliant ISO 8802-3 (IEEE/ANSI 802.3) station. The MENDEC provides the encoding function for data to be transmitted on the network using the high accuracy on-board oscillator, driven by either the crystal oscillator or an external CMOS level compatible clock. The MENDEC also provides the decoding function from data received from the network. The MENDEC contains a Power On Reset (POR) circuit, which ensures that all analog portions of the PCnet-PCI controller are forced into their correct state during power up, and prevents erroneous data transmission and/or reception during this time.

External Crystal Characteristics

When using a crystal to drive the oscillator, the following crystal specification may be used to ensure less than ±0.5 ns jitter at DO±. See Table 4 below.

Table 4. Crystal Specification

Parameter	Min	Nom	Max	Unit
1. Parallel Resonant Frequency		20		MHz
2. Resonant Frequency Error	–50		+50	PPM
3. Change in Resonant Frequency With Respect to Temperature (0 – 70°C)*	–40		+40	PPM
4. Crystal Load Capacitance	20		50	pF
5. Motional Crystal Capacitance (C1)		0.022		pF
6. Series Resistance			35	Ω
7. Shunt Capacitance			7	pF
8. Drive Level			TBD	mW

** Requires trimming specification; not trim is 50 PPM total.*

External Clock Drive Characteristics

When driving the oscillator from a CMOS level external clock source, XTAL2 must be left floating (unconnected). An external clock having the following characteristics must be used to ensure less than ±0.5 ns jitter at DO±. See Table 5.

Table 5. Clock Drive Characteristics

Clock Frequency:	20 MHz ±0.01%
Rise/Fall Time (t_R/t_F):	<= 6 ns from 0.5 V to V_{DD} –0.5 V
XTAL1 HIGH/LOW Time (tHIGH/tLOW):	20 ns min
XTAL1 Falling Edge to Falling Edge Jitter:	< ±0.2 ns at 2.5 V input ($V_{DD/2}$)

MENDEC Transmit Path

The transmit section encodes separate clock and NRZ data input signals into a standard Manchester encoded serial bit stream. The transmit outputs (DO±) are designed to operate into terminated transmission lines. When operating into a 78 Ω terminated transmission line, the transmit signaling meets the required output levels and skew for Cheapernet, Ethernet and IEEE-802.3.

Transmitter Timing and Operation

A 20 MHz fundamental mode crystal oscillator provides the basic timing reference for the MENDEC portion of the PCnet-PCI controller. The crystal is divided by two, to create the internal transmit clock reference. Both clocks are fed into the MENDECs Manchester Encoder to generate the transitions in the encoded data stream. The internal transmit clock is used by the MENDEC to internally synchronize the Internal Transmit Data (ITXDAT) from the controller and Internal Transmit Enable (ITXEN). The internal transmit clock is also used as a stable bit rate clock by the receive section of the MENDEC and controller.

The oscillator requires an external 0.01% timing reference. The accuracy requirements, if an external crystal is used are tighter because allowance for the on-board parasitics must be made to deliver a final accuracy of 0.01%.

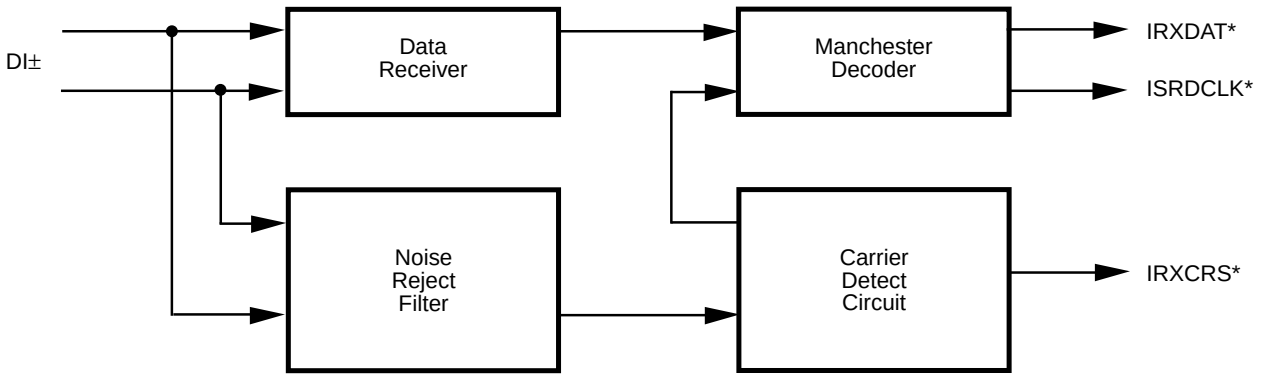
Transmission is enabled by the controller. As long as the ITXEN request remains active, the serial output of the controller will be Manchester encoded and appear at DO±. When the internal request is dropped by the controller, the differential transmit outputs go to one of two idle states, dependent on TSEL in the Mode Register (CSR15, bit 9):

TSEL LOW:	The idle state of DO± yields "ZERO" differential to operate transformer-coupled loads.
TSEL HIGH:	In this idle state, DO+ is positive with respect to DO– (logical HIGH).

Receiver Path

The principal functions of the Receiver are to signal the PCnet-PCI controller that there is information on the receive pair, and separate the incoming Manchester encoded data stream into clock and NRZ data.

The Receiver section (see Receiver Block Diagram) consists of two parallel paths. The receive data path is a ZERO threshold, wide bandwidth line receiver. The carrier path is an offset threshold bandpass detecting line receiver. Both receivers share common bias networks to allow operation over a wide input common mode range.



**Internal signal*

18220C-26

Figure 24. Receiver Block Diagram

Input Signal Conditioning

Transient noise pulses at the input data stream are rejected by the Noise Rejection Filter. Pulse width rejection is proportional to transmit data rate.

The Carrier Detection circuitry detects the presence of an incoming data frame by discerning and rejecting noise from expected Manchester data, and controls the stop and start of the phase-lock loop during clock acquisition. Clock acquisition requires a valid Manchester bit pattern of 1010b to lock onto the incoming message.

When input amplitude and pulse width conditions are met at DI±, the internal enable signal from the MENDEC to controller (IRXCRS) is asserted and a clock acquisition cycle is initiated.

Clock Acquisition

When there is no activity at DI± (receiver is idle), the receive oscillator is phase locked to internal transmit clock. The first negative clock transition (bit cell center of first valid Manchester "0") after IRXCRS is asserted interrupts the receive oscillator. The oscillator is then restarted at the second Manchester "0" (bit time 4) and is phase locked to it. As a result, the MENDEC acquires the clock from the incoming Manchester bit pattern in 4 bit times with a 1010b Manchester bit pattern.

ISRDCLK and IRXDAT are enabled 1/4 bit time after clock acquisition in bit cell 5. IRXDAT is at a HIGH state when the receiver is idle (no ISRDCLK). IRXDAT however, is undefined when clock is acquired and may remain HIGH or change to LOW state whenever ISRDCLK is enabled. At 1/4 bit time through bit cell 5, the controller portion of the PCnet-PCI controller sees the first ISRDCLK transition. This also strobes in the incoming fifth bit to the MENDEC as Manchester "1". IRXDAT may make a transition after the ISRDCLK rising edge in bit cell 5, but its state is still undefined. The Manchester "1" at bit 5 is clocked to IRXDAT output at 1/4 bit time in bit cell 6.

PLL Tracking

After clock acquisition, the phase-locked clock is compared to the incoming transition at the bit cell center (BCC) and the resulting phase error is applied to a correction circuit. This circuit ensures that the phase-locked clock remains locked on the received signal. Individual bit cell phase corrections of the Voltage Controlled Oscillator (VCO) are limited to 100% of the phase

difference between BCC and phase-locked clock. Hence, input data jitter is reduced in ISRDCLK by 10 to 1.

Carrier Tracking and End of Message

The carrier detection circuit monitors the DI± inputs after IRXCRS is asserted for an end of message. IRXCRS de-asserts 1 to 2 bit times after the last positive transition on the incoming message. This initiates the end of reception cycle. The time delay from the last rising edge of the message to IRXCRS de-assert allows the last bit to be strobed by ISRDCLK and transferred to the controller section, but prevents any extra bit(s) at the end of message.

Data Decoding

The data receiver is a comparator with clocked output to minimize noise sensitivity to the DI± inputs. Input error is less than ±35 mV to minimize sensitivity to input rise and fall time. ISRDCLK strobes the data receiver output at 1/4 bit time to determine the value of the Manchester bit, and clocks the data out on IRXDAT on the following ISRDCLK. The data receiver also generates the signal used for phase detector comparison to the internal MENDEC voltage controlled oscillator (VCO).

Differential Input Terminations

The differential input for the Manchester data (DI±) is externally terminated by two 40.2 Ω ±1% resistors and one optional common-mode bypass capacitor, as shown in the Differential Input Termination diagram below. The differential input impedance, Z_{IDF}, and the common-mode input impedance, Z_{ICM}, are specified so that the Ethernet specification for cable termination impedance is met using standard 1% resistor terminators. If SIP devices are used, 39 Ω is also a suitable value. The CI± differential inputs are terminated in exactly the same way as the DI± pair.

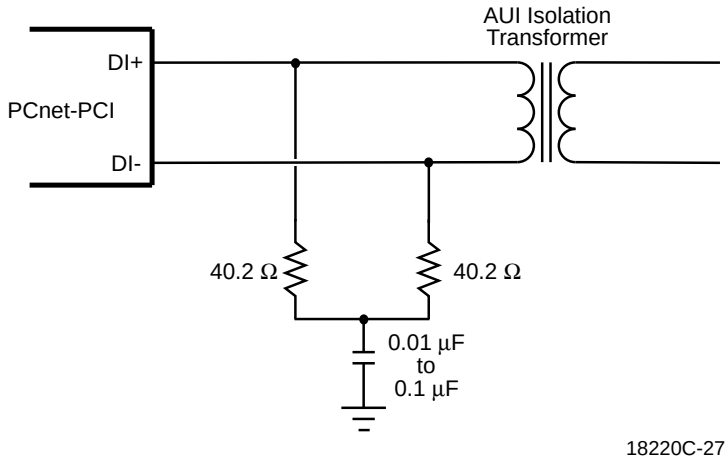


Figure 25. Differential Input Termination

Collision Detection

A MAU detects the collision condition on the network and generates a differential signal at the CI± inputs. This collision signal passes through an input stage which detects signal levels and pulse duration. When the signal is detected by the MENDEC it sets the ICLSN line HIGH. The condition continues for approximately 1.5 bit times after the last LOW-to-HIGH transition on CI±.

Jitter Tolerance Definition

The MENDEC utilizes a clock capture circuit to align its internal data strobe with an incoming bit stream. The clock acquisition circuitry requires four valid bits with the values 1010b. Clock is phase-locked to the negative transition at the bit cell center of the second "0" in the pattern.

Since data is strobed at 1/4 bit time, Manchester transitions which shift from their nominal placement through 1/4 bit time will result in improperly decoded data. With this as the criteria for an error, a definition of Jitter Handling is:

> The peak deviation approaching or crossing 1/4 bit cell position from nominal input transition, for which the MENDEC section will properly decode data.

Attachment Unit Interface (AUI)

The AUI is the PLS (Physical Layer Signaling) to PMA (Physical Medium Attachment) interface which effectively connects the DTE to a MAU. The differential interface provided by the PCnet-PCI controller is fully compliant to Section 7 of ISO 8802-3 (ANSI/IEEE 802.3).

After the PCnet-PCI controller initiates a transmission it will expect to see data "looped-back" on the DI± pair (when the AUI port is selected). This will internally generate a "carrier sense", indicating that the integrity of the data path to and from the MAU is intact, and that the MAU is operating correctly. This "carrier sense" signal must be asserted within TBD bit times after the first transmitted bit on DO± (when using the AUI port). If "carrier sense" does not become active in response to the data transmission, or becomes inactive before the end of transmission, the loss of carrier (LCAR) error bit will be set in the Transmit Descriptor Ring (TMD2, bit 27) after the frame has been transmitted.

Twisted-Pair Transceiver (T-MAU)

The T-MAU implements the Medium Attachment Unit (MAU) functions for the Twisted-Pair Medium, as specified by the supplement to ISO 8802-3 (IEEE/ANSI 802.3) standard (Type 10BASE-T). The T-MAU provides twisted pair driver and receiver circuits, including on-board transmit digital predistortion and receiver squelch and a number of additional features including Link Status indication, Automatic Twisted-Pair Receive Polarity Detection/Correction and Indication, Receive Carrier Sense, Transmit Active and Collision Present indication.

Twisted-Pair Transmit Function

The differential driver circuitry in the TXD± and TXP± pins provides the necessary electrical driving capability and the pre-distortion control for transmitting signals over maximum length Twisted-Pair cable, as specified by the 10BASE-T supplement to the ISO 8802-3 (IEEE/ANSI 802.3) Standard. The transmit function for data output meets the propagation delays and jitter specified by the standard.

Twisted-Pair Receive Function

The receiver complies with the receiver specifications of the ISO 8802-3 (IEEE/ANSI 802.3) 10BASE-T Standard, including noise immunity and received signal rejection criteria ('Smart Squelch'). Signals meeting this criteria appearing at the RXD± differential input pair are routed to the MENDEC. The receiver function meets the propagation delays and jitter requirements specified by the standard. The receiver squelch level drops to half its threshold value after unsquelch to allow reception of minimum amplitude signals and to offset carrier fade in the event of worst case signal attenuation and crosstalk noise conditions.

Note that the 10BASE-T Standard defines the receive input amplitude at the external Media Dependent Interface (MDI). Filter and transformer loss are not specified. The T-MAU receiver squelch levels are defined to account for a 1 dB insertion loss at 10 MHz, which is typical for the type of receive filters/transformers employed.

Normal 10BASE-T compatible receive thresholds are employed when the LRT bit (CSR15[9]) is LOW. When the LRT bit is set (HIGH), the Low Receive Threshold option is invoked, and the sensitivity of the T-MAU receiver is increased. This allows longer line lengths to be employed, exceeding the 100 m target distance of normal 10BASE-T (assuming typical 24 AWG cable). The increased receiver sensitivity compensates for the increased signal attenuation caused by the additional cable distance.

However, making the receiver more sensitive means that it is also more susceptible to extraneous noise, primarily caused by coupling from co-resident services (crosstalk). For this reason, it is recommended that when using the Low Receive Threshold option that the service should be installed on 4-pair cable only. Multipair cables within the same outer sheath have lower crosstalk attenuation, and may allow noise emitted from adjacent pairs to couple into the receive pair, and be of sufficient amplitude to falsely unsquelch the T-MAU.

Link Test Function

The link test function is implemented as specified by 10BASE-T standard. During periods of transmit pair

inactivity, 'Link beat pulses' will be periodically sent over the twisted pair medium to constantly monitor medium integrity.

When the link test function is enabled (DLNKTST bit in CSR15 is cleared), the absence of link beat pulses and receive data on the RXD± pair will cause the TMAU to go into a link fail state. In the link fail state, data transmission, data reception, data loopback and the collision detection functions are disabled, and remain disabled until valid data or >5 consecutive link pulses appear on the RXD± pair. During link fail, the Link Status ($\overline{\text{LNKST}}$ pin) signal is inactive. When the link is identified as functional, the Link Status signal is asserted. The $\overline{\text{LNKST}}$ pin displays the Link Status signal by default.

Transmission attempts during Link Fail state will produce no network activity and will produce LCAR and CERR error indications.

In order to inter-operate with systems which do not implement Link Test, this function can be disabled by setting the DLNKTST bit in CSR15. With link test disabled, the data driver, receiver and loopback functions as well as collision detection remain enabled irrespective of the presence or absence of data or link pulses on the RXD± pair. Link Test pulses continue to be sent regardless of the state of the DLNKTST bit.

Polarity Detection and Reversal

The T-MAU receive function includes the ability to invert the polarity of the signals appearing at the RXD± pair if the polarity of the received signal is reversed (such as in the case of a wiring error). This feature allows data frames received from a reverse wired RXD± input pair to be corrected in the T-MAU prior to transfer to the MENDEC. The polarity detection function is activated following H_RESET or Link Fail, and will reverse the receive polarity based on both the polarity of any previous link beat pulses and the polarity of subsequent frames with a valid End Transmit Delimiter (ETD).

When in the Link Fail state, the T-MAU will recognize link beat pulses of either positive or negative polarity. Exit from the Link Fail state is made due to the reception of 5 – 6 consecutive link beat pulses of identical polarity. On entry to the Link Pass state, the polarity of the last 5 link beat pulses is used to determine the initial receive polarity configuration and the receiver is reconfigured to subsequently recognize only link beat pulses of the previously recognized polarity.

Positive link beat pulses are defined as received signal with a positive amplitude greater than 585 mV (LRT = HIGH) with a pulse width of 60 ns – 200 ns. This positive excursion may be followed by a negative excursion. This definition is consistent with the expected received signal at a correctly wired receiver, when a link beat pulse which fits the template of Figure 14-12 of the 10BASE-T Standard is generated at a transmitter and passed through 100 m of twisted pair cable.

Negative link beat pulses are defined as received signals with a negative amplitude greater than 585 mV with a pulse width of 60 ns – 200 ns. This negative excursion may be followed by a positive excursion. This definition is consistent with the expected received signal at a reverse wired receiver, when a link beat pulse which fits the template of Figure 14-12 in the 10BASE-T Standard is generated at a transmitter and passed through 100 m of twisted pair cable.

The polarity detection/correction algorithm will remain "armed" until two consecutive frames with valid ETD of identical polarity are detected. When "armed", the receiver is capable of changing the initial or previous polarity configuration based on the ETD polarity.

On receipt of the first frame with valid ETD following H_RESET or link fail, the T-MAU will utilize the inferred polarity information to configure its RXD± input, regardless of its previous state. On receipt of a second frame with a valid ETD with correct polarity, the detection/correction algorithm will "lock-in" the received polarity. If the second (or subsequent) frame is not detected as confirming the previous polarity decision, the most recently detected ETD polarity will be used as the default. Note that frames with invalid ETD have no effect on updating the previous polarity decision. Once two consecutive frames with valid ETD have been received, the T-MAU will disable the detection/correction algorithm until either a Link Fail condition occurs or H_RESET is activated.

During polarity reversal, an internal POL signal will be active. During normal polarity conditions, this internal POL signal is inactive. The state of this signal can be read by software and/or displayed by LED when enabled by the LED control bits in the Bus Configuration Registers (BCR4–BCR7).

Twisted-Pair Interface Status

Three signals (XMT, RCV and COL) indicate whether the T-MAU is transmitting, receiving, or in a collision state with both functions active simultaneously. These signals are internal signals and the behavior of the LED outputs depends on how the LED output circuitry is programmed.

The T-MAU will power up in the Link Fail state and normal algorithm will apply to allow it to enter the Link Pass state. In the Link Pass state, transmit or receive activity will be indicated by assertion of RCV signal going active. If T-MAU is selected using the PORTSEL bits in CSR15, then when moving from AUI to T-MAU selection the T-MAU will be forced into the LINK Fail state.

In the Link Fail state, XMT, RCV and COL are inactive.

Collision Detect Function

Activity on both twisted pair signals RXD± and TXD± constitutes a collision, thereby causing the COL signal to be activated. (COL is used by the LED control circuits) COL will remain active until one of the two colliding signals changes from active to idle. However, transmission attempt in Link Fail state results in LCAR and CERR indication. COL stays active for 2 bit times at the end of a collision.

Signal Quality Error (SQE) Test (Heartbeat) Function

The SQE function is disabled when the 10BASE-T port is selected.

Jabber Function

The Jabber function inhibits the twisted pair transmit function of the T-MAU TXD± is active for an excessive period (20 ms – 150 ms). This prevents any one node from disrupting the network due to a 'stuck-on' or faulty transmitter. If this maximum transmit time is exceeded, the T-MAU transmitter circuitry is disabled, the JAB bit is set (CSR4, bit 1) the COL signal is asserted. Once the transmit data stream to the T-MAU is removed, an "unjab" time of 250 ms – 750 ms will elapse before the T-MAU COL and re-enables the transmit circuitry.

Power Down

The T-MAU circuitry can be made to go into power savings mode. This feature is useful in battery powered or low duty cycle systems. The T-MAU will go into power down mode when H_RESET is active, coma mode is active, or the T-MAU is not selected. Refer to the Power Savings Modes section for descriptions of the various power down modes.

Any of the three conditions listed above resets the internal logic of the T-MAU and places the device into power down mode. In this mode, the Twisted-Pair driver pins (TXD±, TXP±) are driven LOW, and the internal T-MAU status signals ($\overline{\text{LNKST}}$, RCVPOL, XMT, RCV and COL) signals are inactive.

Once H_RESET ends, coma mode is disabled, and the T-MAU is selected. The T-MAU will remain in the reset state for up to 10 µs. Immediately after the reset condition is removed, the T-MAU will be forced into the Link Fail state. The T-MAU will move to the Link Pass state only after 5 – 6 link beat pulses and/or a single received message is detected on the RD± pair.

In snooze mode, the T-MAU receive circuitry will remain enabled even while the $\overline{\text{SLEEP}}$ pin is driven LOW.

The T-MAU circuitry will always go into power down mode if H_RESET is asserted, coma mode is enabled, or the T-MAU is not selected.

10BASE-T Interface Connection

Figure 26 shows the proper 10BASE-T network interface design. Refer to the *PCnet Family Technical Manual* (PID #18216A) for more design details, and refer to Appendix A for a list of compatible 10BASE-T filter/ transformer modules.

Note that the recommended resistor values and filter and transformer modules are the same as those used by the IMR (Am79C980) and the IMR+ (Am79C981).

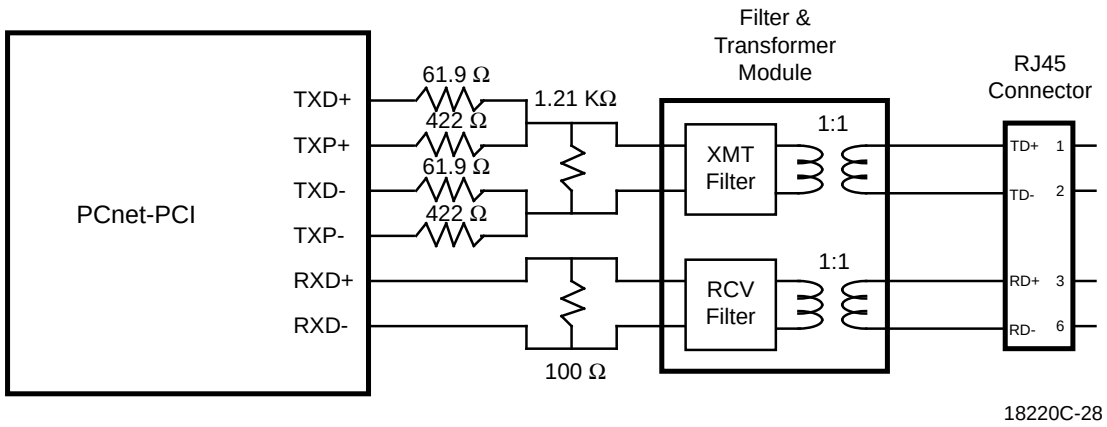


Figure 26. 10BASE-T Interface Connection

Power Savings Modes

The PCnet-PCI controller supports two hardware power savings modes. Both are entered by driving the $\overline{\text{SLEEP}}$ pin LOW.

The PCI interface section is not effected by $\overline{\text{SLEEP}}$. In particular, access to the PCI configuration space remains possible. None of the configuration registers will be reset by $\overline{\text{SLEEP}}$. All I/O accesses to the PCnet-PCI controller will result in a PCI target abort response.

The first power saving mode is called coma mode. In coma mode, the PCnet-PCI controller has no means to use the network to automatically wake itself up. Coma mode is enabled when the AWAKE bit in BCR2 is reset. Coma mode is the default power down mode.

The second power saving mode is called snooze mode. In snooze mode, enabled by setting the AWAKE bit in BCR2 and driving the $\overline{\text{SLEEP}}$ pin LOW, the T-MAU receive circuitry will remain enabled even while the $\overline{\text{SLEEP}}$ pin is driven LOW. The $\overline{\text{LNKST}}$ output is the only one of the LED pins that continues to function. The LNKSTE bit must be set in BCR4 to enable indication of a good 10BASE-T link if there are link beat pulses or valid frames present. This $\overline{\text{LNKST}}$ pin can be used to drive an LED and/or external hardware that directly controls the $\overline{\text{SLEEP}}$ pin of the PCnet-PCI controller. This configuration effectively wakes the system when there is any activity on the 10BASE-T link. Snooze mode can be used only if the T-MAU is the selected network port. Link beat pulses are not transmitted during snooze mode.

If the $\overline{\text{REQ}}$ output is active when the $\overline{\text{SLEEP}}$ pin is asserted, then the PCnet-PCI controller will wait until the $\overline{\text{GNT}}$ input is asserted. Next, the PCnet-PCI controller will deassert the $\overline{\text{REQ}}$ pin and finally, it will internally enter either the coma or snooze sleep mode.

Before the sleep mode is invoked, the PCnet-PCI controller will perform an internal S_RESET. This S_RESET operation will not affect the values of the BCR registers or the PCI configuration space.

The $\overline{\text{SLEEP}}$ pin should not be asserted during power supply ramp-up. If it is desired that $\overline{\text{SLEEP}}$ be asserted at power up time, then the system must delay the assertion of $\overline{\text{SLEEP}}$ until three CLK cycles after the completion of a valid pin $\overline{\text{RST}}$ operation.

Software Access

Configuration Registers

The PCnet-PCI controller supports the 64-byte header portion of the configuration space as predefined by the PCI specification revision 2.0. None of the device specific registers in locations 64 – 255 are used. The layout of the configuration registers in the header region is shown in the table below. All registers required to identify the PCnet-PCI controller and its function are implemented. Additional registers are used to setup the configuration of the PCnet-PCI controller in a system.

31 24	23 16	15 8	7 0	Offset
Device ID		Vendor ID		00h
Status		Command		04h
Base-Class	Sub-Class	Programming IF	Revision ID	08h
Reserved	Header Type	Latency Timer	Reserved	0Ch
Base Address				10h
Reserved				14h
Reserved				18h
Reserved				1Ch
Reserved				20h
Reserved				24h
Reserved				28h
Reserved				2Ch
Reserved				30h
Reserved				34h
Reserved				38h
Reserved	Reserved	Interrupt Pin	Interrupt Line	3Ch

The configuration registers are accessible only by PCI configuration cycles. They can be accessed right after the PCnet-PCI controller is powered-on, even if the read operation of the serial EEPROM is still on-going. All multi-byte numeric fields follow little endian byte ordering. The Command register is the only register cleared by H_RESET. S_RESET as well as asserting $\overline{\text{SLEEP}}$ have no effect on the value of the PCI configuration registers. All write accesses to Reserved locations have no effect, reads from these locations will return a data value of ZERO.

When the PCnet-PCI controller samples its IDSEL input asserted during a configuration cycle, it will acknowledge the cycle by asserting its $\overline{\text{DEVSEL}}$ output. The content of AD[31:00] during the address phase of the configuration cycles must meet the format as shown below:

31 11	10 8	7	6	5 2	1	0
Don't Care	Don't Care	0	0	DWORD Index	0	0

AD[1:0] must both be ZEROs, since the PCnet-PCI controller is not a bridge device. It only recognizes configuration cycles of Type 0 (as defined by the PCI specification revision 2.0). AD[7:2] specify the selected DWORD in the configuration space. AD[7:6] must both be ZERO, since the PCnet-PCI controller does not implement any of the device specific registers in locations 64 – 255. Since AD[1:0] and AD[7:6] must all be ZERO, the lower 8 bits of the address for a configuration cycle are equal to the offset of the DWORD counting from the beginning of the PCI configuration space. AD[10:8] specify one of eight possible functions of a PCI device. The PCnet-PCI controller is a single function device, as indicated in the Header Type register of its PCI configuration space (bit 7, FUNCT = 0). Therefore, the PCnet-PCI controller ignores AD[10:8] during the address phase of a configuration cycle. AD[31:11] are typically used to generate the IDSEL signal. The PCnet-PCI controller ignores all upper address bits.

PCI configuration registers can be accessed with 8-bit, 16-bit or 32-bit transfers. The active bytes within a DWORD are determined by the byte enable signals. E.G. a read of the Sub-Class register can be performed by reading from offset 08h with only $\overline{\text{BE}}$2 being active.

I/O Resources

PCnet-PCI Controller I/O Resource Mapping

The PCnet-PCI controller has several I/O resources. These resources use 32 bytes of I/O space that begin at the PCnet-PCI controller I/O base address.

The PCnet-PCI controller allows two modes of slave access. Word I/O mode treats all PCnet-PCI controller I/O Resources as two-byte entities spaced at two-byte address intervals. Double Word I/O mode treats all PCnet-PCI controller I/O Resources as four-byte entities spaced at four-byte address intervals. The selection of WIO or DWIO mode is accomplished by one of two ways:

- H_RESET function.
- Automatic determination of DWIO mode due to DWORD (double-word) I/O write access to offset 10h.

The PCnet-PCI controller I/O mode setting will default to WIO after H_RESET (i.e. DWIO = 0).

Software may invoke DWIO mode by performing a Double Word write access to the I/O location at offset 10h (RDP). Note that even though the I/O resource mapping changes when the I/O mode setting changes, the RDP location offset is the same for both modes.

1-, 2- and 3-byte accesses to PCnet-PCI controller I/O resources are not allowed during DWIO mode.

The mapping of the PCnet-PCI controller resources into the 32-byte I/O space varies depending upon the setting of the DWIO bit of BCR18. Depending upon the setting of this variable, the 32-byte I/O space will be either Word I/O mapped (WIO) or Double Word I/O mapped (DWIO). A DWIO setting of 0 produces Word I/O mode, while a DWIO setting of 1 produces Double Word I/O mapping.

DWIO is automatically programmed as active when the system attempts a DWORD write access to offset 10h of the PCnet-PCI controller I/O space. The power up reset value of DWIO will be ZERO, and this value will be maintained until a DWORD access is performed to PCnet-PCI controller I/O space.

Therefore, if DWIO mode is desired, it is imperative that the first access to the PCnet-PCI controller be a DWORD write access to offset 10h.

Alternatively, if DWIO mode is not desired, then it is imperative that the software never executes a DWORD write access to offset 10h of the PCnet-PCI controller I/O space.

Once the DWIO bit has been set to a ONE, only a hardware H_RESET can reset it to a ZERO.

The DWIO mode setting is unaffected by the S_RESET or setting the STOP bit.

WIO I/O Resource Map

When the PCnet-PCI controller I/O space is mapped as Word I/O, then the resources that are allotted to the PCnet-PCI controller occur on word boundaries that are offset from the PCnet-PCI controller I/O base address as shown in the following table:

Offset	No. of Bytes	Register
0h	2	APROM
2h	2	APROM
4h	2	APROM
6h	2	APROM
8h	2	APROM
Ah	2	APROM
Ch	2	APROM
Eh	2	APROM
10h	2	RDP
12h	2	RAP (shared by RDP and BDP)
14h	2	Reset Register
16h	2	BDP
18h	2	Vendor Specific Word
1Ah	2	Reserved
1Ch	2	Reserved
1Eh	2	Reserved

When PCnet-PCI controller I/O space is Word mapped, all I/O resources fall on word boundaries and all I/O resources are word quantities. However, while in Word I/O mode, APROM locations may also be accessed as individual bytes either on odd or even byte addresses.

Attempts to write to any PCnet-PCI controller I/O resources *(except to offset 10h, RDP)* as 32 bit quantities while in Word I/O mode are illegal and may cause unexpected reprogramming of the PCnet-PCI controller control registers. Attempts to *read* from any PCnet-PCI controller I/O resources as 32-bit quantities while in Word I/O mode are illegal and will yield undefined values.

An attempt to write to offset 10h (RDP) as a 32 bit quantity while in Word I/O mode will cause the PCnet-PCI controller to exit WIO mode and immediately thereafter, to enter DWIO mode.

Word accesses to non word address boundaries are not allowed while in WIO mode. (A write access may cause unexpected reprogramming of the PCnet-PCI controller control registers. A read access will yield undefined values.)

Accesses of non word quantities to any I/O resource are not allowed while in WIO mode, with the exception of a read to APROM locations. (A write access may cause unexpected reprogramming of the PCnet-PCI controller control registers; a read access will yield undefined values.)

The Vendor Specific Word (VSW) is not implemented by the PCnet-PCI controller. This particular I/O address is reserved for customer use and will not be used by future AMD Ethernet controller products.

DWIO I/O Resource Map

When the PCnet-PCI controller I/O space is mapped as Double Word I/O, then all of the resources that are allotted to the PCnet-PCI controller occur on DWORD boundaries that are offset from the PCnet-PCI controller I/O base address as shown in the table below:

Offset	No. of Bytes	Register
0h	4	APROM
4h	4	APROM
8h	4	APROM
Ch	4	APROM
10h	4	RDP
14h	4	RAP (shared by RDP and BDP)
18h	4	Reset Register
1Ch	4	BDP

When PCnet-PCI I/O space is Double Word mapped, all I/O resources fall on DWORD boundaries. APROM resources are DWORD quantities in DWIO mode. RDP, RAP and BDP contain only two bytes of valid data; the other two bytes of these resources are reserved for future use. (Note that CSR88 is an exception to this rule.) The reserved bits must be written as ZEROs, and when read, are considered undefined.

Accesses to non-doubleword address boundaries are not allowed while in DWIO mode. (A write access may cause unexpected reprogramming of the PCnet-PCI controller control registers; a read access will yield undefined values.)

Accesses of less than 4 bytes to any I/O resource are not allowed while in DWIO mode. (A write access may cause unexpected reprogramming of the PCnet-PCI controller control registers; a read access will yield undefined values.)

A DWORD write access to the RDP offset of 10h will automatically program DWIO mode.

Note that in all cases when I/O resource width is defined as 32 bits, the upper 16 bits of the I/O resource is reserved and written as ZEROS and read as undefined, except for the APROM locations and CSR88.

DWIO mode is exited by asserting the $\overline{RST}$ pin. Assertion of S_RESET or setting the STOP bit of CSR0 will have no effect on the DWIO mode setting.

I/O Space Comments

The following statements apply to both WIO and DWIO mapping:

The RAP is shared by the RDP and the BDP.

The PCnet-PCI controller does not respond to any addresses outside of the offset range 0h–17h when DWIO = 0 or 0h–1Fh when DWIO = 1. I/O offsets 18h through 1Fh are not used by the PCnet-PCI controller when programmed for DWIO = 0 mode; locations 1Ah through 1Fh are reserved for future AMD use and therefore should not be implemented by the user if upward compatibility to future AMD devices is desired.

Note that APROM accesses do not directly access the EEPROM, but are redirected to a set of shadow registers on board the PCnet-PCI controller that contain a copy of the EEPROM contents that was obtained during the automatic EEPROM read operation that follows the H_RESET operation.

PCnet-PCI Controller I/O Base Address

The PCI Configuration Space Base Address register defines what I/O base address the PCnet-PCI controller uses. This register is typically programmed by the PCI configuration utility after system power-up. The PCI configuration utility must also set the IOEN bit in the COMMAND register to enable I/O accesses to the PCnet-PCI controller.

The content of the PCnet-PCI I/O Base Address Registers (BCR16 and BCR17) are ignored.

I/O Register Access

All I/O resources are accessed with similar I/O bus cycles.

I/O accesses to the PCnet-PCI controller begin with a valid $\overline{FRAME}$ signal, the C/$\overline{BE}$[3:0] lines signaling an I/O read or I/O write operation and an address on the AD[31:00] lines that falls within the I/O space of the PCnet-PCI controller. The PCnet-PCI I/O space will be determined by the Base Address Register in the PCI Configuration Space.

The PCnet-PCI controller will respond to an access to its I/O space by asserting the $\overline{DEVSEL}$ signal and eventually, by asserting the $\overline{TRDY}$ signal.

Typical I/O access times are 6 or 7 clock cycles.

APROM Access

The APROM space is a convenient place to store the value of the 48-bit IEEE station address. This space is automatically loaded from the serial EEPROM, if an EEPROM is present. It can be overwritten by the host computer. Its contents have no effect on the operation of the controller. The software must copy the station address from the APROM space to the initialization block or to CSR12-14 in order for the receiver to accept unicast frames directed to this station.

When programmed for WIO mode, any byte or word address from an offset of 0h to an offset of Fh may be read. An appropriate byte or word of APROM contents will be delivered by the PCnet-PCI controller in response to accesses that fall within the APROM range of 0h to Fh.

When programmed for DWIO mode, only DWORD addresses from an offset of 0h to an offset of Fh may be read. An appropriate DWORD of APROM contents will be delivered in response to accesses that fall within the APROM range of 0h to Fh.

Accesses of non-DWORD *quantities* are not allowed in DWIO mode, even though such an access may be properly aligned to a DWORD address boundary.

Write access to any of the APROM locations is allowed, but only 4 bytes on DWORD boundaries in DWIO mode or 2 bytes on word boundaries in WIO mode (only read accesses to the APROM locations can be in 8-bit quantities while in WIO mode). The IESRWE bit (see BCR2) must be set in order to enable a write operation to the APROM. Only the PCnet-PCI controller on-board IEEE Shadow registers are modified by writes to APROM locations. The EEPROM is unaffected by writes to APROM locations.

Note that the APROM locations occupy 16 bytes of space, yet the IEEE station address requirement is for 6 bytes. The 6 bytes of IEEE station address occupy the first 6 locations of the APROM space. The next six bytes are reserved. Bytes 12 and 13 should match the value of the checksum of bytes 1 through 11 and 14 and 15. Bytes 14 and 15 should each be ASCII W (57h). The above requirements must be met in order to be compatible with AMD driver software.

RDP Access (CSR Register Space)

RDP = Register Data Port. The RDP is used with the RAP to gain access to any of the PCnet-PCI controller CSR locations.

Access to any of the CSR locations of the PCnet-PCI controller is performed through the PCnet-PCI controllers Register Data Port (RDP). In order to access a particular CSR location, the Register Address Port (RAP) should first be written with the appropriate CSR address. The RDP now points to the selected CSR. A read of the RDP will yield the selected CSRs data. A write to the RDP will write to the selected CSR.

When programmed for WIO mode, the RDP has a width of 16 bits, hence, all CSR locations have 16 bits of width. Note that when accessing RDP, the upper two bytes of the data bus will be undefined since the byte masks will not be active for those bytes.

If DWIO mode has been invoked, then the RDP has a width of 32 bits, hence, all CSR locations have 32 bits of width and the upper two bytes of the data bus will be active, as indicated by the byte mask. In this case, note that the upper 16 bits of all CSR locations (except CSR88) are reserved and written as ZEROs and read as undefined values. Therefore, during RDP write operations in DWIO mode, the upper 16 bits of all CSR locations should be written as ZEROs.

RAP Access

RAP = Register Address Port. The RAP is used with the RDP and with the BDP to gain access to any of the CSR and BCR register locations, respectively. The RAP contains the address pointer that will be used by an access to either the RDP or BDP. Therefore, it is necessary to set the RAP value before accessing a specific CSR or BCR location. Once the RAP has been written with a value, the RAP value remains unchanged until another RAP write occurs, or until an H_RESET or S_RESET occurs. RAP is set to all ZEROs when an H_RESET or S_RESET occurs. RAP is unaffected by the STOP bit.

When programmed for WIO mode, the RAP has a width of 16 bits. Note that when accessing RAP, the lower two bytes of the data bus will be undefined since the byte masks will not be active for those bytes

When programmed for DWIO mode, the RAP has a width of 32 bits. In DWIO mode, the upper 16 bits of the RAP are reserved and written as ZEROs and read as undefined. These bits should be written as ZEROs.

BDP Access (BCR Register Space)

BDP = Bus Configuration Register Data Port. The BDP is used with the RAP to gain access to any of the PCnet-PCI controller BCR locations.

Access to any of the BCR locations of the PCnet-PCI controller is performed through the PCnet-PCI controllers BCR Data Port (BDP); in order to access a particular BCR location, the Register Address Port (RAP) should first be written with the appropriate BCR address. The BDP now points to the selected BCR. A read of the BDP will yield the selected BCR s data. A write to the BDP will write to the selected BCR.

When programmed for WIO mode, the BDP has a width of 16 bits, hence, all BCR locations have 16 bits of width in WIO mode. Note that when operating in WIO mode,

the upper two bytes of the data bus will be undefined since the byte mask will not be active for those bytes.

If DWIO mode has been invoked, then the BDP has a width of 32 bits, hence, all BCR locations have 32 bits of width and the upper two bytes of the data bus will be active, as indicated by the byte mask. In this case, note that the upper 16 bits of all BCR locations are reserved and written as ZEROs and read as undefined. Therefore, during BDP write operations in DWIO mode, the upper 16 bits of all BCR locations should be written as ZEROs.

RESET Register (S_RESET)

A read of the reset register creates an internal S_RESET pulse in the PCnet-PCI controller. This read access cycle must be 16 bits wide in WIO mode and 32 bits wide in DWIO mode. The internal S_RESET pulse that is generated by this access is different from both the assertion of the hardware $\overline{RST}$ pin (H_RESET) and from the assertion of the software STOP bit. Specifically, the reset registers S_RESET will be the equivalent of the assertion of the $\overline{RST}$ pin (H_RESET) assertion for all CSR locations, but S_RESET will have no effect at all on the BCR or PCI configuration space locations, and S_RESET will not cause a deassertion of the $\overline{REQ}$ pin.

The NE2100 LANCE based family of Ethernet cards requires that a write access to the reset register follows each read access to the reset register. The PCnet-PCI controller does not have a similar requirement. The write access is not required but it does not have any harmful effects.

Write accesses to the reset register will have no effect on the PCnet-PCI controller.

Note that a read access of the reset register will take longer than the normal I/O access time of the PCnet-PCI controller. This is because an internal S_RESET pulse will be generated due to this access, and the access will not be allowed to complete on the system bus until the internal S_RESET operation has been completed. This is to avoid the problem of allowing a new I/O access to proceed while the S_RESET operation has not yet completed, which would result in erroneous data being returned by (or written into) the PCnet-PCI controller. The length of a read of the Reset register can be as long as 64 clock cycles.

Note that a read of the reset register will **not** cause a deassertion of the $\overline{REQ}$ signal, if it happens to be active at the time of the read of the reset register. The $\overline{REQ}$ signal will remain active until the $\overline{GNT}$ signal is asserted. Following the read of the reset register, on the next clock cycle after the $\overline{GNT}$ signal is asserted, the PCnet-PCI controller will deassert the $\overline{REQ}$ signal. No bus master accesses will have been performed during this brief bus ownership period.

Note that this behavior differs from that which occurs following the assertion of a minimum-width pulse on the $\overline{RST}$ pin (H_RESET). A $\overline{RST}$ pin assertion will cause the $\overline{REQ}$ signal to deassert within six clock cycles following the assertion. In the $\overline{RST}$ pin case, the PCnet-PCI controller will not wait for the assertion of the $\overline{GNT}$ signal before deasserting the $\overline{REQ}$ signal.

Vendor Specific Word

This I/O offset is reserved for use by the system designer. The PCnet-PCI controller will not respond to accesses directed toward this offset. The Vendor Specific Word is only available when the PCnet-PCI controller is programmed to word I/O mode (DWIO = 0).

If more than one Vendor Specific Word is needed, it is suggested that the VSW location should be divided into a VSW Register Address Pointer (VSWRAP) at one location (e.g. VSWRAP at byte location 18h or word location 30h, depending upon DWIO state) and a VSW Data Port (VSWDP) at the other location (e.g. VSWDP at byte location 19h or word location 32h, depending upon DWIO state). Alternatively, the system may capture RAP data accesses in parallel with the PCnet-PCI controller and therefore share the PCnet-PCI controller RAP to allow expanded VSW space. PCnet-PCI controller will not respond to access to the VSW I/O address.

Reserved I/O Space

These locations are reserved for future use by AMD. The PCnet-PCI controller does not respond to accesses directed toward these locations, but future AMD products that are intended to be upward compatible with the PCnet-PCI controller device may decode accesses to these locations. Therefore, the system designer may not utilize these I/O locations.

Hardware Access

PCnet-PCI Controller Master Accesses

The PCnet-PCI controller has a bus interface compatible with PCI specification revision 2.0.

Complete descriptions of the signals involved in bus master transactions for each mode may be found in the pin description section of this document. Timing diagrams for master accesses may be found in the block description section for the Bus Interface Unit. This section simply lists the types of master accesses that will be performed by the PCnet-PCI controller with respect to data size and address information.

The PCnet-PCI controller will support master accesses only to 32-bit peripherals. The PCnet-PCI controller does not support master accesses to 8-bit or 16-bit memory. The PCnet-PCI controller is not compatible with 8-bit systems, since there is no mode that supports PCnet-PCI controller accesses to 8-bit peripherals.

Table 6 describes all possible bus master accesses that the PCnet-PCI controller will perform. The right most column lists all operations that may execute the given access:

Table 6. Bus Master Accesses

Access	Mode	$\overline{BE}$[3:0]	Operation
4-byte read	Read	0000	descriptor read or initialization block read or transmit data buffer read
4-byte write	Write	0000	descriptor write or receive data buffer write
3-byte write	Write	1000	receive data buffer write
3-byte write	Write	0001	receive data buffer write
2-byte write	Write	1100	receive data buffer write
2-byte write	Write	1001*	receive data buffer write
2-byte write	Write	0011	receive data buffer write
1-byte write	Write	1110	receive data buffer write
1-byte write	Write	1101*	receive data buffer write
1-byte write	Write	1011*	receive data buffer write
1-byte write	Write	0111	descriptor write or receive data buffer write

** Cases marked with an asterisk represent extreme boundary conditions that are the result of programming one- and two-byte buffer sizes, and therefore will not be seen under normal circumstances.*

Note that all PCnet-PCI controller master read operations will always activate all byte enables. Therefore, no one-, two- or three-byte read operations are indicated in the table.

In the instance where a transmit buffer pointer address begins on a non-DWORD boundary, the pointer will be truncated to the next DWORD boundary address that lies below the given pointer address and the first read access from the transmit buffer will be indicated on the byte enable signals as a four-byte read from this address. Any data from byte lanes that lie outside of the boundary indicated by the buffer pointer will be discarded inside of the PCnet-PCI controller. Similarly, if the end of a transmit buffer occurs on a non-DWORD boundary, then all byte lanes will be indicated as active by the byte enable signals, and any data from byte lanes that lie outside of the boundary indicated by the buffer pointer will be discarded inside of the PCnet-PCI controller.

Slave Access to I/O Resources

The PCnet-PCI device is always a 32-bit peripheral on the system bus. However, the width of individual software resources on board the PCnet-PCI controller may be either 16-bits or 32-bits. The PCnet-PCI controller I/O resource widths are determined by the setting of the DWIO bit as indicated in the following table:

DWIO Setting	PCnet-PCI Controller I/O Resource Width	Example Application
DWIO = 0	16-bit	Existing PCnet-ISA driver that assumes 16-bit I/O mapping and 16-bit resource widths
DWIO = 1	32-bit	New drivers written specifically for the PCnet-PCI controller

Note that when I/O resource width is defined as 32 bits (DWIO mode), the upper 16 bits of the I/O resource is reserved and written as ZEROS and read as undefined, except for the APROM locations and CSR88. The APROM locations and CSR88 are the only I/O resources for which all 32 bits will have defined values. However, this is true only when the PCnet-PCI controller is in DWIO mode.

Configuring the PCnet-PCI controller for DWIO mode is accomplished whenever there is any attempt to perform a 32-bit write access to the RDP location (offset 10h). See the DWIO section for more details.

Table 7 describes all possible bus slave accesses that may be directed toward the PCnet-PCI controller. (i.e., the PCnet-PCI controller is the target device during the transfer.) The first column indicates the type of slave access. RD stands for READ, WR for a WRITE operation. The second column indicates the value of the C/$\overline{BE}$[3:0] lines during the data phase of the transfer. The four byte columns (AD[31:24], AD[23:16], AD[15:8], AD[7:0]) indicate the value on the address/data bus during the data phase of the access. “data” indicates the position of the active bytes; “copy” indicates the positions of copies of the active bytes; “undef” indicates byte locations that are undefined during the transfer.

Table 7. Bus Slave Accesses

TYPE	$\overline{BE}$[3:0]	AD [31:24]	AD [23:16]	AD [15:8]	AD [7:0]	Comments
RD	0000	data	data	data	data	DWORD access to DWORD address, e.g. 300h, 30Ch, 310h (DWIO mode only)
RD	1100	undef	undef	data	data	word access to even word address, e.g. 300h, 30Ch, 310h (WIO mode only)
RD	0011	data	data	copy	copy	word access to odd word address, e.g. 302h, 30Eh, 312h (WIO mode only)
RD	1110	undef	undef	undef	data	byte access to lower byte of even word address, e.g. 300h, 304h (WIO mode only, APROM accesses only)
RD	1101	undef	undef	data	undef	byte access to upper byte of even word address, e.g. 301h, 305h (WIO mode only, APROM accesses only)
RD	1011	undef	data	undef	copy	byte access to lower byte of odd word address, e.g. 302h, 306h (WIO mode only, APROM accesses only)
RD	0111	data	undef	copy	undef	byte access to upper byte of odd word address, e.g. 303h, 307h (WIO mode only, APROM accesses only)
WR	0000	data	data	data	data	DWORD access to DWORD address, e.g. 300h, 30Ch, 310h (DWIO mode only)
WR	1100	undef	undef	data	data	word access to even word address, e.g. 300h, 30Ch, 310h (WIO mode only)
WR	0011	data	data	undef	undef	word access to odd word address, e.g. 302h, 30Eh, 312h (WIO mode only)

EEPROM Microwire Access

The PCnet-PCI controller contains a built-in capability for reading and writing to an external EEPROM. This built-in capability consists of a Microwire interface for direct connection to a Microwire compatible EEPROM, an automatic EEPROM read feature, and a user-programmable register that allows direct access to the Microwire interface pins.

Automatic EEPROM Read Operation

Shortly after the deassertion of the $\overline{RST}$ pin, the PCnet-PCI controller will read the contents of the EEPROM that is attached to the Microwire interface. Because of this automatic-read capability of the PCnet-PCI controller, an EEPROM can be used to program many of the features of the PCnet-PCI controller at power-up, allowing system-dependent configuration information to be stored in the hardware, instead of inside of operating code.

If an EEPROM exists on the Microwire interface, the PCnet-PCI controller will read the EEPROM contents at the end of the H_RESET operation. The EEPROM contents will be serially shifted into a temporary register and then sent to various register locations on board the PCnet-PCI controller. The host can access the PCI Configuration Space during the EEPROM read operations. Access to the PCnet-PCI I/O resources, however, is not possible during the EEPROM read operation. The PCnet-PCI controller will terminate these I/O accesses with the assertion of $\overline{DEVSEL}$ and $\overline{STOP}$ while $\overline{TRDY}$ is not asserted, signaling to the initiator to retry the access at a later time.

A checksum verification is performed on the data that is read from the EEPROM. If the checksum verification of the EEPROM data fails, then at the end of the EEPROM read sequence, the PCnet-PCI controller will force all EEPROM-programmable BCR registers back to their

H_RESET default values. The content of the APROM locations (offsets 0h – Fh from the I/O base address), however, will not be cleared. The 8-bit checksum for the entire 36 bytes of the EEPROM should be FFh.

If no EEPROM is present at the time of the automatic read operation, then the PCnet-PCI controller will recognize this condition and will abort the automatic read operation and reset both the PREAD and PVALID bits in BCR19. All EEPROM-programmable BCR registers will be assigned their default values after H_RESET. The content of the Address PROM locations (offsets 0h – Fh from the I/O base address) will be undefined.

If the user wishes to modify any of the configuration bits that are contained in the EEPROM, then the seven command, data and status bits of BCR19 can be used to write to the EEPROM. After writing to the EEPROM, the host should set the PREAD bit of BCR19. This action forces a PCnet-PCI controller re-read of the EEPROM so that the new EEPROM contents will be loaded into the EEPROM-programmable registers on board the PCnet-PCI controller. (The EEPROM-programmable registers may also be reprogrammed directly, but only information that is stored in the EEPROM will be preserved at system power-down.) When the PREAD bit of BCR19 is set, it will cause the PCnet-PCI controller to terminate further accesses to internal I/O resources with the PCI retry cycle. Accesses to the PCI configuration space is still possible.

EEPROM Auto-Detection

The PCnet-PCI controller uses the EESK/$\overline{\text{LED1}}$ pin to determine if an EEPROM is present in the system. At all rising CLK edges during the assertion of the $\overline{\text{RST}}$ pin, the PCnet-PCI controller will sample the value of the EESK/$\overline{\text{LED1}}$ pin. If the sampled value is a ONE, then the PCnet-PCI controller assumes that an EEPROM is present, and the EEPROM read operation begins shortly after the $\overline{\text{RST}}$ pin is deasserted. If the sampled value of EESK/$\overline{\text{LED1}}$ is a ZERO, then the PCnet-PCI controller assumes that an external pulldown device is holding the EESK/$\overline{\text{LED1}}$ pin low, and therefore, there is no EEPROM in the system. Note that if the designer creates a system that contains an LED circuit on the EESK/$\overline{\text{LED1}}$ pin but has no EEPROM present, then the EEPROM auto-detection function will incorrectly conclude that an EEPROM is present in the system. However, this will not pose a problem for the PCnet-PCI controller, since it will recognize the lack of an EEPROM at the end of the read operation, when the checksum verification fails.

Systems Without an EEPROM

Some systems may be able to save the cost of an EEPROM by storing the ISO 8802-3 (IEEE/ANSI 802.3) station address and other configuration information somewhere else in the system. There are several design choices:

- If the LED1 is not needed in the system, then the system designer may connect the EESK/$\overline{\text{LED1}}$ pin to a resistive pulldown device. This will indicate to the EEPROM auto-detection function that no EEPROM is present.
- If the LED1 function is needed in the system, then the system designer will connect the EESK/$\overline{\text{LED1}}$ pin to a resistive pullup device and the EEPROM auto-detection function will incorrectly conclude that an EEPROM is present in the system. However, this will not pose a problem for the PCnet-PCI controller, since it will recognize the lack of an EEPROM at the end of the read operation, when the checksum verification fails.

In either case, following the PCI configuration, additional information, including the ISO 8802-3 (IEEE/ANSI 802.3) station address, may be loaded into the PCnet-PCI controller. Note that the IESRWE bit (bit 8 of BCR2) must be set before the PCnet-PCI controller will accept writes to the APROM offsets within the PCnet-PCI I/O resources map. Startup code in the system BIOS can perform the PCI configuration accesses, the IESRWE bit write, and the APROM writes.

Direct Access to the Microwire Interface

The user may directly access the Microwire port through the EEPROM register, BCR19. This register contains bits that can be used to control the Microwire interface pins. By performing an appropriate sequence of I/O accesses to BCR19, the user can effectively write to and read from the EEPROM. This feature may be used by a system configuration utility to program hardware configuration information into the EEPROM.

EEPROM-Programmable Registers

The following registers contain configuration information that will be programmed automatically during the EEPROM read operation:

1. I/O offsets 0h – Fh	APROM locations
2. BCR2	Miscellaneous Configuration register
3. BCR16	not used in the PCnet-PCI controller
4. BCR17	not used in the PCnet-PCI controller
5. BCR18	Burst Size and Bus Control Register
6. BCR21	Not Used

If the PREAD bit (BCR19) is reset to ZERO and the PVALID bit (BCR19) is reset to ZERO, then the EEPROM read has experienced a failure and the contents of the EEPROM programmable BCR register will be set to default H_RESET values. The content of the APROM locations, however, will not be cleared.

Note that accesses to the APROM I/O locations do not directly access the Address EEPROM itself. Instead, these accesses are routed to a set of shadow registers on board the PCnet-PCI controller that are loaded with a copy of the EEPROM contents during the automatic read operation that immediately follows the H_RESET operation.

EEPROM MAP

The automatic EEPROM read operation will access 18 words (i.e. 36 bytes) of the EEPROM. The format of the EEPROM contents is shown in Table 8, beginning with the byte that resides at the lowest EEPROM address:

Table 8. EEPROM Contents

EEPROM WORD Address	EEPROM Contents			
	Byte Addr.	Most Significant Byte	Byte Addr.	Least Significant Byte
00h (lowest EEPROM address)	01h	2nd byte of the ISO 8802-3 (IEEE/ANSI 802.3) station physical address for this node	00h	first byte of the ISO 8802-3 (IEEE/ANSI 802.3) station physical address for this node, where “first byte” refers to the first byte to appear on the 802.3 medium
01h	03h	4th byte of the node address	02h	3rd byte of the node address
02h	05h	6th byte of the node address	04h	5th byte of the node address
03h	07h	reserved location: must be 00h	06h	reserved location must be 00h
04h	09h	Hardware ID: must be 11h if compatibility to AMD drivers is desired	08h	reserved location must be 00h
05h	0Bh	user programmable space	0Ah	user programmable space
06h	0Dh	MSByte of two-byte checksum, which is the sum of bytes 00h–0Bh and bytes 0Eh and 0Fh	0Ch	LSByte of two-byte checksum, which is the sum of bytes 00h–0Bh and bytes 0Eh and 0Fh
07h	0Fh	must be ASCII “W” (57h) if compatibility to AMD driver software is desired	0Eh	must be ASCII “W” (57h) if compatibility to AMD driver software is desired
08h	11h	BCR16[15:8] (not used)	10h	BCR16[7:0] (not used)
09h	13h	BCR17[15:8] (not used)	12h	BCR17[7:0] (not used)
0Ah	15h	BCR18[15:8] (Burst Size and Bus Control)	14h	BCR18[7:0] (Burst Size and Bus Control)
0Bh	17h	BCR2[15:8] (Misc. configuration)	16h	BCR2[7:0] (Misc. configuration)
0Ch	19h	BCR21[15:8] (Not Used)	18h	BCR21[7:0] (Not Used)
0Dh	1Bh	reserved location must be 00h	1Ah	reserved location must be 00h
0Eh	1Dh	reserved location must be 00h	1Ch	reserved location must be 00h
0Fh	1Fh	checksum adjust byte for the first 36 bytes of the EEPROM contents; checksum of the first 36 bytes of the EEPROM should total to FFh	1Eh	reserved location must be 00h
10h	21h	reserved location must be 00h	20h	reserved location must be 00h
11h	23h	user programmable byte locations	22h	user programmable byte locations

Note that the first bit out of any WORD location in the EEPROM is treated as the MSB of the register that is being programmed. For example, the first bit out of EEPROM WORD location 08h will be written into BCR16[15], the second bit out of EEPROM WORD location 08h will be written into BCR16[14], etc.

There are two checksum locations within the EEPROM. The first is required for the EEPROM address. This checksum will be used by AMD driver software to verify that the ISO 8802-3 (IEEE/ANSI 802.3) station address has not been corrupted. The value of bytes 0Ch and 0Dh should match the sum of bytes 00h through 0Bh and 0Eh and 0Fh. The second checksum location – byte 1Fh – is not a checksum total, but is, instead, a checksum adjustment. The value of this byte should be such that the total checksum for the entire 36 bytes of EEPROM data equals the value FFh. The checksum adjust byte is needed by the PCnet-PCI controller in order to verify that the EEPROM contents have not been corrupted.

Transmit Operation

The transmit operation and features of the PCnet-PCI controller are controlled by programmable options. The PCnet-PCI controller offers a136-byte Transmit FIFO to provide frame buffering for increased system latency, automatic retransmission with no FIFO reload, and automatic transmit padding.

Transmit Function Programming

Automatic transmit features such as retry on collision, FCS generation/transmission, and pad field insertion can all be programmed to provide flexibility in the (re-)transmission of messages.

Disable retry on collision (DRTY) is controlled by the DRTY bit of the Mode register (CSR15) in the initialization block.

Automatic pad field insertion is controlled by the APAD_XMT bit in CSR4. If APAD_XMT is set, automatic pad field insertion is enabled, the DXMTFCS feature is over-ridden, and the 4-byte FCS will be added to the transmitted frame unconditionally. If APAD_XMT is clear, no pad field insertion will take place and runt packet transmission is possible.

The disable FCS generation/transmission feature can be programmed dynamically on a frame by frame basis. See the ADD_FCS description of TMD1.

Transmit FIFO Watermark (XMTFW) in CSR80 sets the point at which the BMU requests more data from the transmit buffers for the FIFO. A minimum of XMTFW empty spaces must be available in the transmit FIFO before the BMU will request the system bus in order to transfer transmit packet data into the transmit FIFO.

Transmit Start Point (XMTSP) in CSR80 sets the point when the transmitter actually attempts to transmit a frame onto the media. A minimum of XMTSP bytes must be written to the transmit FIFO for the current frame before transmission of the current frame will begin. (When automatically padded packets are being sent, it is conceivable that the XMTSP is not reached when all of the data has been transferred to the FIFO. In this case, the transmission will begin when all of the packet data has been placed into the transmit FIFO.)

When the entire frame is in the FIFO, attempts at transmission of preamble will commence regardless of the value in XMTSP. The default value of XMTSP is 10b, meaning there has to be 64 bytes in the Transmit FIFO to start a transmission.

Automatic Pad Generation

Transmit frames can be automatically padded to extend them to 64 data bytes (excluding preamble). This allows the minimum frame size of 64 bytes (512 bits) for 802.3/Ethernet to be guaranteed with no software intervention from the host/controlling process.

Setting the APAD_XMT bit in CSR4 enables the automatic padding feature. The pad is placed between the LLC data field and FCS field in the 802.3 frame.FCS is always added if the frame is padded, regardless of the state of DXMTFCS. The transmit frame will be padded by bytes with the value of 00h. The default value of APAD_XMT is 0; this will disable auto pad generation after H_RESET.

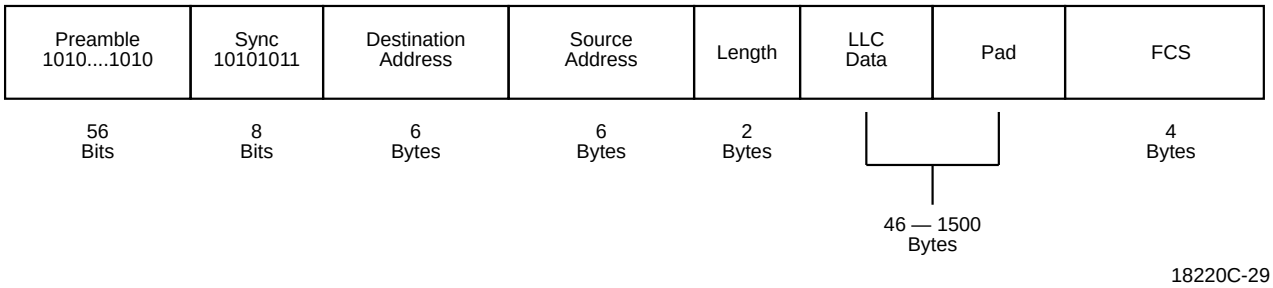


Figure 27. ISO 8802-3(IEEE/ANSI 802.3) Data Frame

It is the responsibility of upper layer software to correctly define the actual length field contained in the message to correspond to the total number of LLC Data bytes encapsulated in the packet (length field as defined in the ISO 8802-3 (IEEE/ANSI 802.3) standard). The length value contained in the message is not used by the PCnet-PCI controller to compute the actual number of pad bytes to be inserted. The PCnet-PCI controller will append pad bytes dependent on the actual number of bits transmitted onto the network. Once the last data byte of the frame has completed, prior to appending the FCS, the PCnet-PCI controller will check to ensure that 544 bits have been transmitted. If not, pad bytes are added to extend the frame size to this value, and the FCS is then added.

The 544 bit count is derived from the following:

Minimum frame size (excluding preamble, including FCS)	64	bytes	512	bits
Preamble/SFD size	8	bytes	64	bits
FCS size	4	bytes	32	bits

To be classed as a minimum size frame at the receiver, the transmitted frame must contain:

Preamble + (Min Frame Size + FCS) bits

At the point that FCS is to be appended, the transmitted frame should contain:

Preamble + (Min Frame Size – FCS) bits

64 + (512 – 32) bits

A minimum length transmit frame from the PCnet-PCI controller will therefore be 576 bits, after the FCS is appended.

The Ethernet specification assumes that minimum length messages will be at least 64 bytes in length.

Transmit FCS Generation

Automatic generation and transmission of FCS for a transmit frame depends on the value of DXMTFCS bit in CSR15. When DXMTFCS = 0 the transmitter will generate and append the FCS to the transmitted frame. If the automatic padding feature is invoked (APAD_XMT is set in CSR4), the FCS will be appended by the PCnet-PCI controller regardless of the state of DXMTFCS. Note that the calculated FCS is transmitted most significant bit first. The default value of DXMTFCS is 0 after H_RESET.

Transmit Exception Conditions

Exception conditions for frame transmission fall into two distinct categories. Those which are the result of normal network operation, and those which occur due to abnormal network and/or host related events.

Normal events which may occur and which are handled autonomously by the PCnet-PCI controller include collisions within the slot time with automatic retry. The PCnet-PCI controller will ensure that collisions which occur within 512 bit times from the start of transmission (including preamble) will be automatically retried with no host intervention. The transmit FIFO ensures this by guaranteeing that data contained within the FIFO will not be overwritten until at least 64 bytes (512 bits) of preamble plus address, length and data fields have been transmitted onto the network without encountering a collision.

If 16 total attempts (initial attempt plus 15 retries) fail, the PCnet-PCI controller sets the RTRY bit in the current transmit TDTE in host memory (TMD2), gives up ownership (resets the OWN bit to ZERO) for this frame, and processes the next frame in the transmit ring for transmission.

Abnormal network conditions include:

- Loss of carrier.
- Late collision.
- SQE Test Error. (does not apply to 10BASE-T port)

These should not occur on a correctly configured 802.3 network, and will be reported if they do.

When an error occurs in the middle of a multi-buffer frame transmission, the error status will be written in the current descriptor. The OWN bit(s) in the subsequent descriptor(s) will be reset until the STP (the next frame) is found.

Loss of Carrier

A loss of carrier condition will be reported if the PCnet-PCI controller cannot observe receive activity whilst it is transmitting on the AUI port. After the PCnet-PCI controller initiates a transmission it will expect to see data "looped-back" on the DI± pair. This will internally generate a "carrier sense", indicating that the integrity of the data path to and from the MAU is intact, and that the MAU is operating correctly. This "carrier sense" signal must be asserted about 6 bit times before the last transmitted bit on DO±. If "carrier sense" does not become active in response to the data transmission, or becomes inactive before the end of transmission, the loss of carrier (LCAR) error bit will be set in TMD2 after the frame has been transmitted. The frame will not be re-tried on the basis of an LCAR error.

When the 10BASE-T port is selected, LCAR will be reported for every packet transmitted during the Link fail condition.

Late Collision

A late collision will be reported if a collision condition occurs after one slot time (512 bit times) after the transmit process was initiated (first bit of preamble commenced). The PCnet-PCI controller will abandon the transmit process for the particular frame, set Late Collision (LCOL) in the associated TMD2, and process the next transmit frame in the ring. Frames experiencing a late collision will not be re-tried. Recovery from this condition must be performed by upper layer software.

SQE Test Error

During the inter packet gap time following the completion of a transmitted message, the AUI CI± pair is asserted by some transceivers as a self-test. The integral Manchester Encoder/Decoder will expect the SQE Test Message (nominal 10 MHz sequence) to be returned via the CI± pair, within a 40 network bit time period after DI±

goes inactive (this does not apply if the 10BASE-T port is selected). If the CI± input is not asserted within the 40 network bit time period following the completion of transmission, then the PCnet-PCI controller will set the CERR bit in CSR0. CERR will be asserted in 10BASE-T mode after transmit if T-MAU is in Link Fail state. CERR will never cause $\overline{\text{INTA}}$ to be activated. It will, however, set the ERR bit CSR0.

Receive Operation

The receive operation and features of the PCnet-PCI controller are controlled by programmable options.

Address Matching

The PCnet-PCI controller supports three types of address matching: unicast, multicast, and broadcast. The normal address matching procedure can be modified by programming three bits in the MODE register (PROM, DRCVBC, and DRCBC).

If the first bit received after the start of frame delimiter (the least significant bit of the first byte of the destination address field) is 0, the frame is unicast, which indicates that the frame is meant to be received by a single node. If the first bit received is 1, the frame is multicast, which indicates that the frame is meant to be received by a group of nodes. If the destination address field contains all ones, the frame is broadcast, which is a special type of multicast. Frames with the broadcast address in the destination address field are meant to be received by all nodes on the local area network.

When a unicast frame arrives at the PCnet-PCI controller, the controller will accept the frame if the destination address field of the incoming frame exactly matches the 6-byte station address stored in the PADR registers (CSR12, CSR13, and CSR14). The byte ordering is such that the first byte received from the network (after the SFD) must match the least significant byte of CSR12 (PADR[7:0]), and the sixth byte received must match the most significant byte of CSR14 (PADR[47:40]).

If DRCVPA (bit 13 in the MODE register) is set. the PCnet-PCI controller will not accept unicast frames.

If the incoming frame is multicast the PCnet-PCI controller performs a calculation on the contents of the destination address field to determine whether or not to accept the frame. This calculation is explained in the section that describes the Logical Address Filter (LADRF).

If all bits of the LADRF registers are 0 no multicast frames are accepted, except for broadcast frames.

Although broadcast frames are classified as special multicast frames, they are treated differently by the PCnet-PCI controller hardware. Broadcast frames are always accepted, except when DRCVBC (bit 14 in the MODE register) is set.

None of the address filtering described above applies when the PCnet-PCI controller is operating in the promiscuous mode. In the promiscuous mode, all properly formed packets are received, regardless of the contents of their destination address fields. The promiscuous mode overrides the Disable Receive Broadcast bit (DRCVBC bit l4 in the MODE register) and the Disable Receive Physical Address bit (DRCVPA, bit 13 MODE register).

The PCnet-PCI controller operates in promiscuous mode when PROM (bit 15 in the MODE register) is set.

Receive Function Programming

Automatic pad field stripping is enabled by setting the ASTRP_RCV bit in CSR4. This can provide flexibility in the reception of messages using the 802.3 frame format.

All receive frames can be accepted by setting the PROM bit in CSR15. When PROM is set, the PCnet-PCI controller will attempt to receive all messages, subject to minimum frame enforcement. Promiscuous mode over rides the effect of the Disable Receive Broadcast bit on receiving broadcast frames.

The point at which the BMU will start to transfer data from the receive FIFO to buffer memory is controlled by the RCVFW bits in CSR80. The default established during H_RESET is 10b which sets the threshold flag at 64 bytes empty.

Automatic Pad Stripping

During reception of an 802.3 frame the pad field can be stripped automatically. ASTRP_RCV (CSR4, bit 0) = 1 enables the automatic pad stripping feature. The pad field will be stripped before the frame is passed to the FIFO, thus preserving FIFO space for additional frames. The FCS field will also be stripped, since it is computed at the transmitting station based on the data and pad field characters, and will be invalid for a receive frame that has had the pad characters stripped.

The number of bytes to be stripped is calculated from the embedded length field (as defined in the ISO 8802-3 (IEEE/ANSI 802.3) definition) contained in the frame. The length indicates the actual number of LLC data bytes contained in the message. Any received frame which contains a length field less than 46 bytes will have the pad field stripped (if ASTRP_RCV is set). Receive frames which have a length field of 46 bytes or greater will be passed to the host unmodified.

Since any valid Ethernet Type field value will always be greater than a normal 802.3 Length field (≥46), the PCnet-PCI controller will not attempt to strip valid Ethernet frames.

Note that for some network protocols, the value passed in the Ethernet Type and/or 802.3 Length field is not compliant with either standard and may cause problems.

Figure 28 shows the byte/bit ordering of the received length field for an 802.3 compatible frame format.

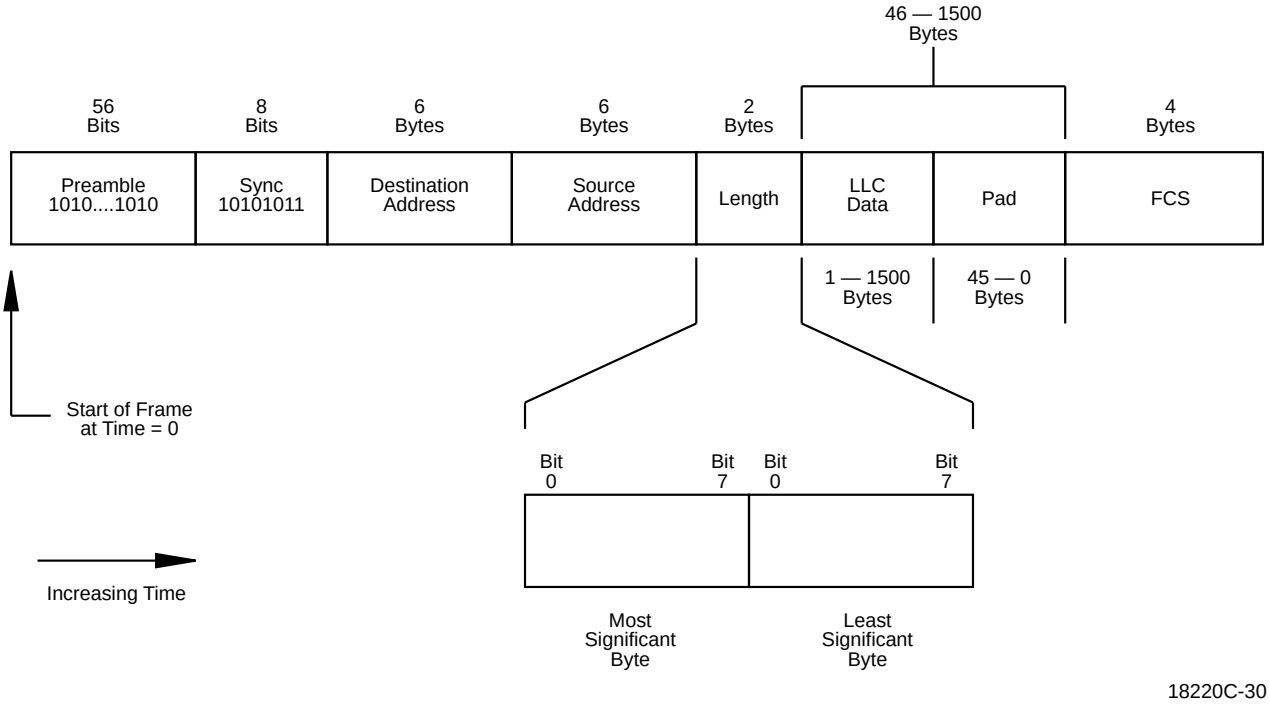


Figure 28. 802.3 Frame and Length Field Transmission Order

Receive FCS Checking

Reception and checking of the received FCS is performed automatically by the PCnet-PCI controller. Note that if the Automatic Pad Stripping feature is enabled, the FCS for padded frames will be verified against the value computed for the incoming bit stream including pad characters, but the FCS value for a padded frame will not be passed to the host. If an FCS error is detected in any frame, the error will be reported in the CRC bit in RMD1.

Receive Exception Conditions

Exception conditions for frame reception fall into two distinct categories; those which are the result of normal network operation, and those which occur due to abnormal network and/or host related events.

Normal events which may occur and which are handled autonomously by the PCnet-PCI controller are basically collisions within the slot time and automatic runt packet rejection. The PCnet-PCI controller will ensure that collisions which occur within 512 bit times from the start of reception (excluding preamble) will be automatically deleted from the receive FIFO with no host intervention. The receive FIFO will delete any frame which is composed of fewer than 64 bytes provided that the Runt Packet Accept (RPA bit in CSR124) feature has not been enabled. This criterion will be met regardless of whether the receive frame was the first (or only) frame in the FIFO or if the receive frame was queued behind a previously received message.

Abnormal network conditions include:

- FCS errors
- Late Collision

Host related receive exception conditions include MISS, BUFF, and OFLO. These are described in the BMU section.

Loopback Operation

Loopback is a mode of operation intended for system diagnostics. In this mode, the transmitter and receiver are both operating at the same time so that the controller receives its own transmissions. The controller provides two types of internal loopback and one type of external loopback. In internal loopback mode, the transmitted data can be looped back to the receiver at one of two places inside the controller without actually transmitting any data to the external network. The receiver will move the received data to the next receive buffer, where it can be examined by software. Alternatively, in external loop-

back mode, data can be transmitted to and received from the external network.

There are restrictions on loopback operation. The PCnet-PCI controller has only one FCS generator circuit. The FCS generator can be used by the transmitter to generate the FCS to append to the frame, or it can be used by the receiver to verify the FCS of the received frame. It can not be used by the receiver and transmitter simultaneously.

If the FCS generator is connected to the receiver, the transmitter will not append an FCS to the frame, but the receiver will check for one. The user can, however, calculate the FCS value for a frame and include this four-byte number in the transmit buffer.

If the FCS generator is connected to the transmitter, the transmitter will append an FCS to the frame, but the receiver will not check for the FCS. However, the user can verify the FCS by software.

During loopback, the FCS logic can be allocated to the receiver by setting DXMTFCS = 1 in CSR15.

If DXMTFCS=0, the MAC Engine will calculate and append the FCS to the transmitted message. The receive message passed to the host will therefore contain an additional 4 bytes of FCS. In this loopback configuration, the receive circuitry cannot detect FCS errors if they occur.

If DXMTFCS=1, the last four bytes of the transmit message must contain the (software generated) FCS computed for the transmit data preceding it. The MAC Engine will transmit the data without addition of an FCS field, and the FCS will be calculated and verified at the receiver.

The loopback facilities of the MAC Engine allow full operation to be verified without disturbance to the network. Loopback operation is also affected by the state of the Loopback Control bits (LOOP, MENDECL, and INTL) in CSR15. This affects whether the internal MENDEC is considered part of the internal or external loopback path.

The multicast address detection logic uses the FCS generator circuit. Therefore, in the loopback mode(s), the multicast address detection feature of the MAC Engine, programmed by the contents of the Logical Address Filter (LADRF [63:0] in CSRs 8–11) can only be tested when DXMTFCS=1, allocating the FCS generator to the receiver. All other features operate identically in loopback as in normal operation, such as automatic transmit padding and receive pad stripping.

When performing an internal loopback, no frame will be transmitted to the network. However, when the PCnet-PCI controller is configured for internal loopback the receiver will not be able to detect network traffic. External loopback tests will transmit frames onto the network if the AUI port is selected, and the PCnet-PCI controller will receive network traffic while configured for external loopback when the AUI port is selected. Runt Packet Accept is automatically enabled when any loopback mode is invoked.

Loopback mode can be performed with any frame size. Runt Packet Accept is internally enabled (RPA bit in CSR124 is not affected) when any loopback mode is invoked. This is to be backwards compatible to the LANCE (Am7990) software.

When the 10BASE-T MAU is selected in external loopback mode, the collision detection is disabled. This is necessary, because a collision in a 10BASE-T system is defined as activity on the transmitter outputs and receiver inputs at the same time, which is exactly what occurs during external loopback.

Since a 10BASE-T hub does not normally feed the station's transmitter outputs back into the station's receiver inputs, the use of external loopback in a 10BASE-T system usually requires some sort of external hardware that connects the outputs of the 10BASE-T MAU to its inputs.

LED Support

The PCnet-PCI controller can support up to 3 LEDs.

LED outputs $\overline{\text{LNKST}}$ and $\overline{\text{LED1}}$ allow for direct connection of an LED and its supporting pullup device. LED output $\overline{\text{LED3}}$ may require an additional buffer between the PCnet-PCI controller output pin and the LED and its supporting pullup device.

Because the $\overline{\text{LED3}}$ output is multiplexed with other PCnet-PCI controller functions, it may not always be possible to connect an LED circuit directly to the $\overline{\text{LED3}}$ pin. For example, when an LED circuit is directly connected to the EEDO/$\overline{\text{LED3}}$ pin, then it is not possible for most serial EEPROM devices to sink enough I_{OL} to maintain a valid low level on the EEDO input to the PCnet-PCI controller. Therefore, in applications that require both an EEPROM and a third LED, then it is necessary to buffer the $\overline{\text{LED3}}$ circuit from the EEPROM-PCnet-PCI connection. The LED registers in the BCR resource space allow each LED output to be programmed for either active high or active low operation, so that both inverting and non-inverting buffering choices are possible.

In applications where an EEPROM is not needed, the $\overline{\text{LED3}}$ pin may be directly connected to an LED circuit. The PCnet-PCI $\overline{\text{LED3}}$ pin driver will be able to sink enough current to properly drive the LED circuit.

By default, after H_RESET, the 3 LED outputs are configured in the following manner:

LED output	Default Interpretation	Default Drive Enable	Default Output Polarity
$\overline{\text{LNKST}}$	Link Status	Enabled	Active LOW
$\overline{\text{LED1}}$	Receive	Enabled	Active LOW
$\overline{\text{LED3}}$	Transmit	Enabled	Active LOW

For each LED register, each of the status signals is ANDed with its enable signal, and these signals are all ORed together to form a combined status signal. Each LED pins combined status signal runs to a pulse stretcher, which consists of a 3-bit shift register clocked at 38 Hz (26 ms). The data input of each shift register is normally at logic 0. The OR gate output for each LED register asynchronously sets all three bits of its shift register when the output becomes asserted. The inverted output of each shift register is used to control an LED pin. Thus the pulse stretcher provides 2–3 clocks of stretched LED output, or 52 ms to 78 ms.

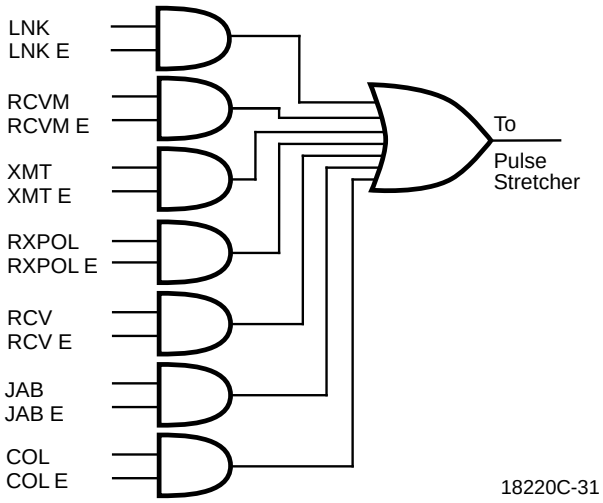


Figure 29. LED Control Logic

The diagram above shows the LED signal circuit that exists for each LED pin within the PCnet-PCI controller.

H_RESET, S_RESET, and STOP

There are three different types of RESET operations that may be performed on the PCnet-PCI device, H_RESET, S_RESET and STOP. These names have been used throughout the document. The following is a description of each type of RESET operation:

H_RESET

H_RESET= HARDWARE_RESET is a PCnet-PCI RESET operation that has been created by the proper assertion of the $\overline{\text{RST}}$ PIN of the PCnet-PCI device. When the minimum pulse width timing as specified in the $\overline{\text{RST}}$ pin description has been satisfied, then an internal RESET operation will be performed.

H_RESET will RESET all of or some portions of CSR0, 3, 4, 15, 58, 80, 82, 100, 112, 114, 122, 124 and 126 to default values. H_RESET will RESET all of or some portions of BCR 2, 4, 5, 6, 7, 18, 19, 20, 21 to default values. H_RESET will reset the Command register in the PCI configuration space. H_RESET will cause the microcode program to jump to its RESET state. Following the end of the H_RESET operation, the PCnet-PCI controller will attempt to read the EEPROM device through the EEPROM Microwire interface. H_RESET resets the T-MAU into the link fail state.

S_RESET

S_RESET = SOFTWARE_RESET is a PCnet-PCI RESET operation that has been created by a read access to the RESET REGISTER which is located at offset 14hex from the PCnet-PCI I/O base address.

S_RESET will RESET all of or some portions of CSR0, 3, 4, 15, 80, 100 and 124 to default values. S_RESET will not affect any of the BCR and PCI configuration space locations. S_RESET will cause the microcode program to jump to its RESET state. Following the end of the S_RESET operation, the PCnet-PCI controller will NOT attempt to read the EEPROM device. S_RESET sets the T-MAU into the link fail state.

Note that S_RESET will not cause a deassertion of the $\overline{\text{REQ}}$ signal, if it happens to be active at the time of the read to the reset register. The $\overline{\text{REQ}}$ signal will remain active until the $\overline{\text{GNT}}$ signal is asserted. Following the read of the RESET register, on the next clock cycle after the $\overline{\text{GNT}}$ signal is asserted, the PCnet-PCI controller will deassert the $\overline{\text{REQ}}$ signal. No bus master accesses will have been performed during this brief bus ownership period.

STOP

STOP is a PCnet-PCI RESET operation that has been created by the ASSERTION of the STOP bit in CSR0. That is, a STOP RESET is generated by writing a ONE to the STOP bit of CSR0 when the STOP bit currently has a value of ZERO. If the STOP bit value is currently a ONE and a ONE is rewritten to the STOP bit, then NO STOP RESET will be generated.

STOP will RESET all or some portions of CSR0, 3, and 4 to default values. STOP will not affect any of the BCR and PCI configuration space locations. STOP will cause the microcode program to jump to its RESET state. Following the end of the STOP operation, the PCnet-PCI controller will NOT attempt to read the EEPROM device. For the identity of individual CSRs and bit locations that are affected by STOP, see the individual CSR register descriptions. Setting the STOP bit does not affect the T-MAU.

NAND Tree Testing

The PCnet-PCI controller provides a NAND tree test mode to allow checking connectivity to the device on a printed circuit board. The NAND tree is built on all PCI bus signals.

NAND tree testing is enabled by asserting $\overline{RST}$. All PCI bus signals will become inputs on the asserting of $\overline{RST}$. The result of the NAND tree test can be observed on the NOUT pin.

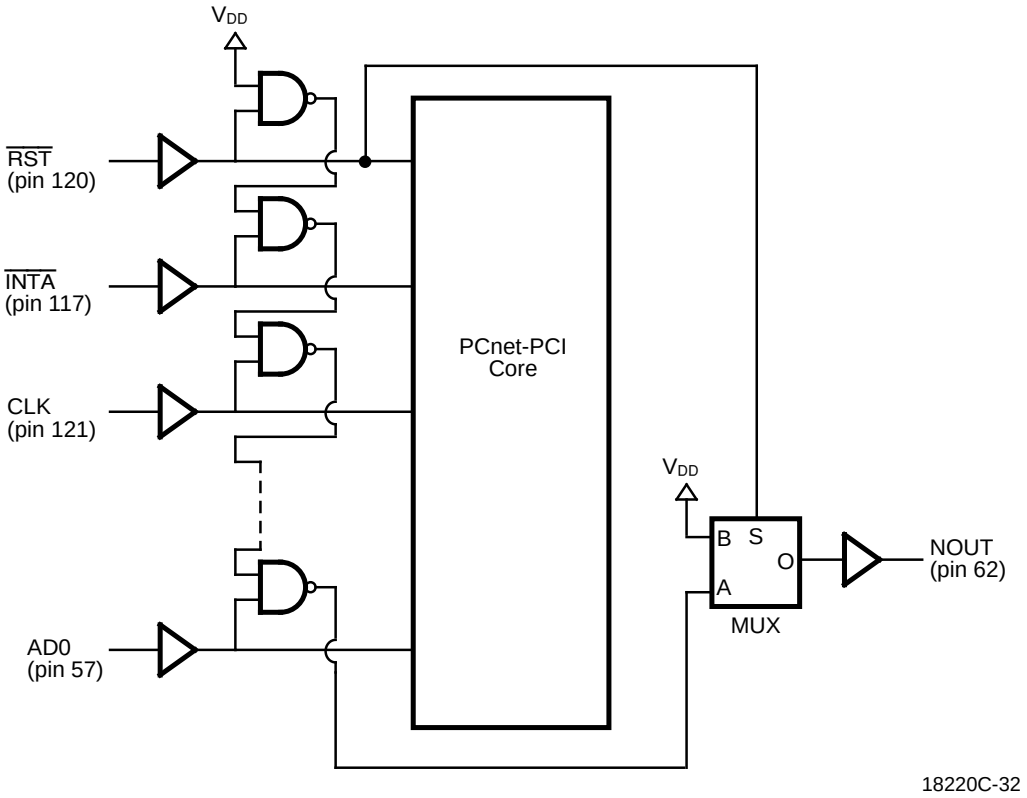


Figure 30. NAND Tree

Pin 120 ($\overline{RST}$) is the first input to the NAND tree. Pin 117 ($\overline{INTA}$) is the second input to the NAND tree, followed by pin 121 (CLK). All other PCI bus signals follow, counter-clockwise, with pin 57 (AD0) being the last. Pins labeled NC and all power supply pins are not part of the NAND tree. Table 9 shows the complete list of pins connected to the NAND tree.

Table 9. NAND Tree Configuration

NAND Tree Input #	Pin #	Name	NAND Tree Input #	Pin #	Name	NAND Tree Input #	Pin #	Name
1	120	$\overline{RST}$	18	15	AD21	35	36	AD15
2	117	$\overline{INTA}$	19	16	AD20	36	38	AD14
3	121	CLK	20	18	AD19	37	39	AD13
4	123	$\overline{GNT}$	21	19	AD18	38	40	AD12
5	126	$\overline{REQ}$	22	21	AD17	39	41	AD11
6	128	AD31	23	22	AD16	40	42	AD10
7	129	AD30	24	23	C/$\overline{BE}$2	41	44	AD9
8	131	AD29	25	24	$\overline{FRAME}$	42	45	AD8
9	132	AD28	26	25	$\overline{IRDY}$	43	47	C/$\overline{BE}$0
10	2	AD27	27	26	$\overline{TRDY}$	44	48	AD7
11	3	AD26	28	27	$\overline{DEVSEL}$	45	49	AD6
12	5	AD25	29	28	$\overline{STOP}$	46	51	AD5
13	6	AD24	30	29	$\overline{LOCK}$	47	52	AD4
14	7	C/$\overline{BE}$3	31	31	$\overline{PERR}$	48	53	AD3
15	10	IDSEL	32	32	$\overline{SERR}$	49	54	AD2
16	12	AD23	33	34	PAR	50	56	AD1
17	13	AD22	34	35	C/$\overline{BE}$1	51	57	AD0

$\overline{RST}$ must be asserted low to start a NAND tree test sequence. Initially, all NAND tree inputs except $\overline{RST}$ should be driven high. This will result in a high output at the NOUT pin. If the NAND tree inputs are driven from high to low in the same order as they are connected to build the NAND tree, NOUT will toggle every time an additional input is driven low. NOUT will change to a ZERO, when $\overline{INTA}$ is driven low and all other NAND tree inputs stay high. NOUT will toggle back to high, when CLK is additionally driven low. The square wave will continue until all NAND tree inputs are driven low. NOUT will be high, when all NAND tree inputs are driven low.

Note that some of the pins connected to the NAND tree are outputs in normal mode of operation. They must not be driven from an external source until the PCnet-PCI controller is configured for NAND tree testing.

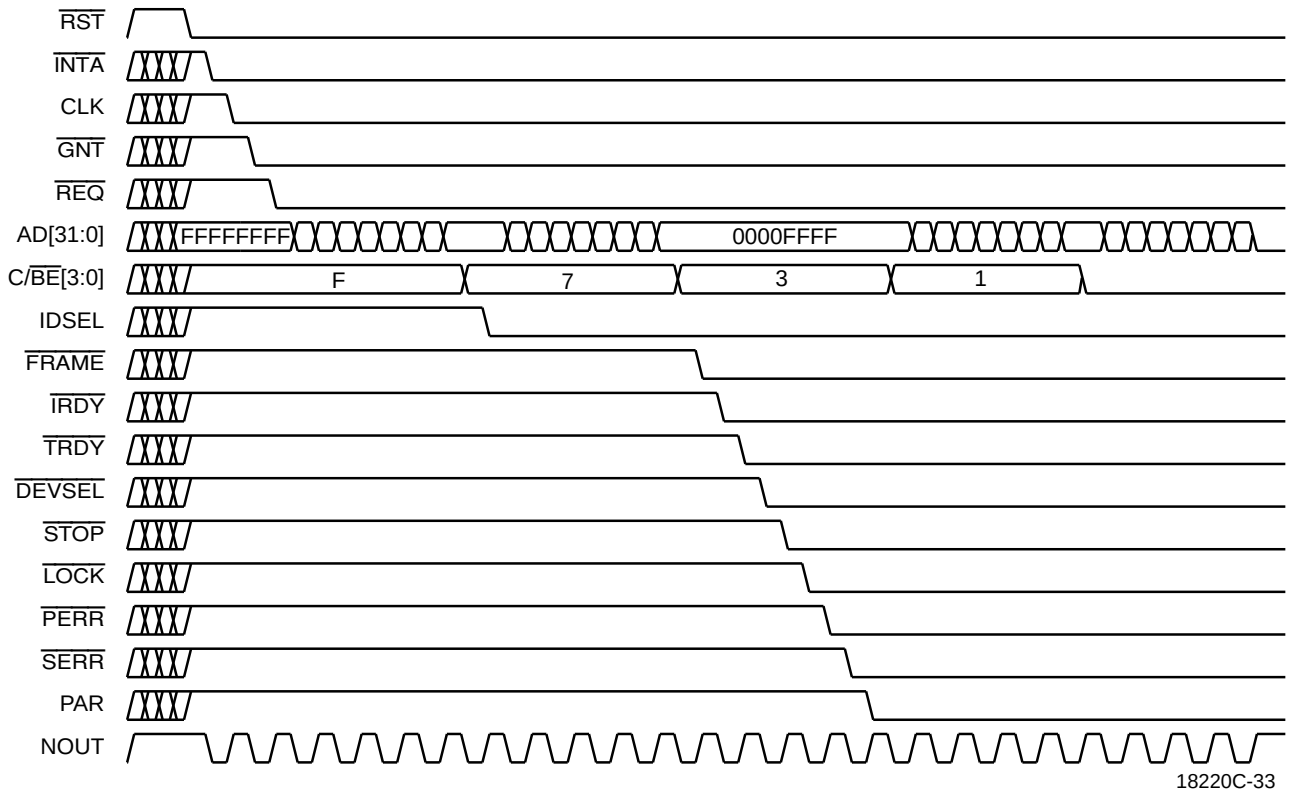


Figure 31. NAND Tree Waveform

USER ACCESSIBLE REGISTERS

The PCnet-PCI controller has three types of user registers: the PCI configuration registers, the Control and Status registers (CSR) and the Bus Control registers (BCR).

The PCnet-PCI controller implements all PCnet-ISA (Am79C960) registers all LANCE (Am7990) registers, all ILACC (Am79C900) registers, plus a number of additional registers. The PCnet-PCI controller CSRs are compatible with both the PCnet-ISA (Am79C960) CSRs and all of the LANCE (Am7990) CSRs upon power up. Compatibility to the ILACC set of CSRs requires one access to the Software Style register (BCR20, bits 7–0) to be performed. By setting an appropriate value of the Software Style register (BCR20, bits 7–0) the user can select a set of CSRs that are compatible with the ILACC set of CSRs.

The PCI configuration registers can be accessed in any data width. All other registers must be accessed according to the IO mode that is currently selected. When WIO mode is selected, all other register locations are defined to be 16 bits in width. When DWIO mode is selected, all these register locations are defined to be 32 bits in width, with the upper 16 bits of most register locations marked as reserved locations with undefined values. When performing register write operations in DWIO mode, the upper 16 bits should always be written as zeros, except for CSR88. When performing register read operations in DWIO mode, the upper 16 bits of I/O resources should always be regarded as having undefined values, except for CSR88.

PCnet-PCI registers can be divided into four groups:

PCI Configuration Registers:

Registers that are intended to be initialized by the system initialization procedure (e.g. BIOS device initialization routine) to program the operation of the PCnet-PCI controller PCI bus interface.

Setup Registers:

Registers that are intended to be initialized by the device driver to program the operation of various PCnet-PCI controller features.

Running Registers:

Registers that are intended to be used by the device driver software once the PCnet-PCI controller is running to access status information and to pass control information.

Test Registers:

Registers that are intended to be used only for testing and diagnostic purposes.

Below is a list of the registers that fall into each of the first three categories. Those registers that are not included in either of these lists can be assumed to be intended for diagnostic purposes.

PCI Configuration Registers

The following is a list of those registers that would typically need to be programmed once during the initialization of the PCnet-PCI controller within a system

- Base Address register
- Interrupt Line register
- Status register
- Command register

Setup Registers

The following is a list of those registers that would typically need to be programmed once during the setup of the PCnet-PCI controller within a system. The control bits in each of these registers typically do not need to be modified once they have been written. However, there are no restrictions as to how many times these registers may actually be accessed. Note that if the default power up values of any of these registers is acceptable to the application, then such registers need never be accessed at all. Also note that some of these registers may be programmable through the EEPROM read operation, and therefore do not necessarily need to be written to by the system initialization procedure or by the driver software.

CSR1	Initialization Address[15:0]
CSR2	Initialization Address[31:16]
CSR3	Interrupt Masks and Deferral Control
CSR4	Test and Features Control
CSR8	Logical Address Filter[15:0]
CSR9	Logical Address Filter[31:16]
CSR10	Logical Address Filter[47:32]
CSR11	Logical Address Filter[63:48]
CSR12	Physical Address Filter[15:0]
CSR13	Physical Address Filter[31:16]
CSR14	Physical Address Filter[47:32]
CSR15	Mode Register
CSR24	Base Address of Receive Ring Lower
CSR25	Base Address of Receive Ring Upper
CSR30	Base Address of Transmit Ring Lower
CSR31	Base Address of Transmit Ring Upper
CSR47	Polling Interval
CSR76	Receive Ring Length
CSR78	Transmit Ring Length
CSR80	DMA Transfer Counter and FIFO Threshold Control
CSR82	Bus Activity Timer
CSR100	Memory Error Timeout Register

CSR122	Receiver Packet Alignment Control
BCR2	Misc. configuration
BCR18	Bus Size and Burst Control Register
BCR20	Software Style

Running Registers

The following is a list of those registers that would typically need to be periodically read and perhaps written during the normal running operation of the PCnet-PCI controller within a system. Each of these registers contains control bits or status bits or both.

RAP	Register Address Port Register
CSR0	PCnet-PCI controller Status Register
CSR4	Test and Features Control
CSR112	Missed Frame Count
CSR114	Receive Collision Count

PCI Configuration Registers

The PCnet-PCI controller supports the 64-byte header portion of the configuration space as defined by the PCI specification revision 2.0. None of the device specific registers in locations 64 – 255 are used. The layout of the configuration registers in the header region is shown in the table below. All registers required to identify the PCnet-PCI controller and its function are implemented. Additional registers are used to setup the configuration of the PCnet-PCI controller in a system.

The configuration registers are accessible only by PCI configuration cycles. They can be accessed right after the PCnet-PCI controller is powered-on, even if the read operation of the serial EEPROM is still on-going. All multi-byte numeric fields follow little endian byte ordering. The Command register is the only register cleared by H_RESET. S_RESET as well as asserting $\overline{\text{SLEEP}}$ have no effect on the value of the PCI configuration registers. All write accesses to Reserved locations have no affect, reads from these locations will return a data value of ZERO.

Vendor ID (Offset 00h)

The Vendor ID register is a 16-bit register that identifies the manufacturer of the PCnet-PCI controller. Advanced Micro Devices, Inc.'s (AMD) Vendor ID is 1022h. Note that this vendor ID is not the same as the Manufacturer ID in CSR88 and CSR89. The vendor ID is assigned by the PCI Special Interest Group.

The Vendor ID register is located at offset 00h in the PCI Configuration Space. It is read only.

Device ID Register (Offset 02h)

The Device ID register is a 16-bit register that uniquely identifies the PCnet-PCI controller within AMD's product line. The PCnet-PCI Device ID is 2000h. Note that this Device ID is not the same as the Part number in CSR88 and CSR89. The Device ID is assigned by Advanced Micro Devices, Inc.

The Device ID register is located at offset 02h in the PCI Configuration Space. It is read only.

31 24	23 16	15 8	7 0	Offset
Device ID		Vendor ID		00h
Status		Command		04h
Base-Class	Sub-Class	Programming IF	Revision ID	08h
Reserved	Header Type	Latency Timer	Reserved	0Ch
Base Address				10h
Reserved				14h
Reserved				18h
Reserved				1Ch
Reserved				20h
Reserved				24h
Reserved				28h
Reserved				2Ch
Reserved				30h
Reserved				34h
Reserved				38h
Reserved	Reserved	Interrupt Pin	Interrupt Line	3Ch

Command Register (offset 04h)

The Command register is a 16-bit register used to control the gross functionality of the PCnet-PCI controller. It controls the PCnet-PCI controller's ability to generate and respond to PCI bus cycles. To logically disconnect the PCnet-PCI device from all PCI bus cycles except Configuration cycles, a value of ZERO should be written to this register.

The Command register is located at offset 04h in the PCI Configuration Space. It is read and written by the host.

15-10 RES — Reserved locations. Read as ZERO, write operations have no effect.

9 FBTBEN — Fast Back-to-Back enable. Read as ZERO, write operations have no effect. The PCnet-PCI controller will not generate Fast Back-to-Back cycles.

8 SERREN — SERR enable. Controls the assertion of the $\overline{SERR}$ pin. $\overline{SERR}$ is disabled when SERREN is cleared. $\overline{SERR}$ will be asserted on detection of an address parity error and if both, SERREN and PERREN (bit 6 of this register) are set.

SERREN is cleared by H_RESET and is not effected by S_RESET or asserting the $\overline{SLEEP}$ pin.

7 ADSTEP — Address/data stepping. Read as ONE, write operations have no effect. The PCnet-PCI controller uses address stepping for the first address phase of each bus master period. $\overline{FRAME}$ will be asserted on the second CLK following the assertion of $\overline{GNT}$ indicating a valid address on the AD bus.

6 PERREN — Parity Error Response enable. Enables the parity error response functions. When PERREN is '0' and the PCnet-PCI controller detects a parity error, it only sets the Detected Parity Error bit in the Status register. When PERREN is '1', the PCnet-PCI controller asserts $\overline{PERR}$ on the detection of a data parity error. It also sets the DATAPERR bit (bit 8 in the Status register), when the data parity error occurred during a master cycle. PERREN also enables reporting address parity errors through the $\overline{SERR}$ pin and the $\overline{SERR}$ bit in the Status register.

PERREN is cleared by H_RESET and is not effected by S_RESET or asserting the $\overline{SLEEP}$ pin.

5 VGASNOOP — VGA palette snoop. Read as ZERO, write operations have no effect.

4 MWIEN — Memory Write and Invalidate Cycle enable. Read as ZERO, write operations have no effect. The PCnet-PCI controller only generates Memory Write cycles.

3 SCYCEN — Special Cycle enable. Read as ZERO, write operations have no effect. The PCnet-PCI controller ignores all Special Cycle operations.

2 BMEN — Bus Master enable. Setting BMEN enables the PCnet-PCI controller to become a bus master on the PCI bus. The host must set BMEN before setting the INIT bit in CSR0 of the PCnet-PCI controller. (Setting INIT causes the PCnet-PCI controller to start its first bus master operation, which is reading in the initialization block.)

BMEN is cleared by H_RESET and is not effected by S_RESET or asserting the $\overline{SLEEP}$ pin.

1 MEMEN — Memory Space access enable. Read as ZERO, write operations have no effect. The PCnet-PCI controller has no memory mapped resources.

0 IOEN — I/O Space access enable. The PCnet-PCI controller will ignore all I/O accesses when IOEN is cleared. The host must set IOEN before the first I/O access to the device. The Base Address register at offset 10h must be programmed with a valid I/O address before setting IOEN.

IOEN is cleared by H_RESET and is not effected by S_RESET or asserting the $\overline{SLEEP}$ pin.

Status Register (Offset 06h)

The Status register is a 16-bit register that contains status information for the PCI bus related events. It is located at offset 06h in the PCI Configuration Space.

15 PERR — Parity Error. PERR is set when the PCnet-PCI controller detects a parity error.

The PCnet-PCI controller samples the AD[31:00], C/$\overline{BE}$[3:0]

and the PAR lines for a parity error at the following times:

- In slave mode, during the address phase of any PCI bus command.
- In slave mode, during the data phase of all I/O and Configuration Write commands that select the PCnet-PCI controller.
- In master mode, during the data phase of all Memory Read and Memory Read Line commands.

During the data phase of the memory write command, the PCnet-PCI controller sets the PERR bit if the target reports a data parity error by asserting the PERR signal.

PERR is not effected by the state of the Parity Error Response enable bit (bit 6 in the Control register).

PERR is set by the PCnet-PCI controller and cleared by writing a ONE. Writing a ZERO has no effect. PERR is not affected by H_RESET or S_RESET or asserting the SLEEP pin.

14 SERR — Signaled SERR. SERR is set when the PCnet-PCI controller detects an address parity error, and both, SERREN and PEREN (bits 8 and 6 of the Command register) are set.

SERR is set by the PCnet-PCI controller and cleared by writing a “1”. Writing a “0” has no effect. SERR is not affected by H_RESET or S_RESET or asserting the SLEEP pin.

13 RMABORT — Received Master Abort. RMABORT is set when the PCnet-PCI controller terminates a master cycle with a master abort sequence.

RMABORT is set by the PCnet-PCI controller and cleared by writing a “1”. Writing a “0” has no effect. RMABORT is not affected by H_RESET or S_RESET or asserting the SLEEP pin.

12 RTABORT — Received Target Abort. RTABORT is set when a target terminates a PCnet-PCI master cycle with a target abort sequence.

RTABORT is set by the PCnet-PCI controller and cleared by writing a “1”. Writing a “0” has no effect. RTABORT is not affected by H_RESET or S_RESET or asserting the SLEEP pin.

11 STABORT — Send Target Abort. STABORT is set when the PCnet-PCI controller terminates a slave access with a target abort sequence.

STABORT is set by the PCnet-PCI controller and cleared by writing a “1”. Writing a “0” has no effect. STABORT is not affected by H_RESET or S_RESET or asserting the SLEEP pin.

10–9 DEVSEL — DEVSEL timing. DEVSEL is set to 01b (medium), indicating the PCnet-PCI controller will assert DEVSEL two CLK periods after FRAME is asserted.

DEVSEL is read only.

8 DATAPERR — Data Parity Error detected. DATAPERR is set when the PCnet-PCI controller detects a data parity error during master mode and the Parity Error Response enable bit (bit 6 in the Control register) is set.

During the data phase of all Memory Read and Memory Read Line commands, the PCnet-PCI controller checks for parity error by sampling the AD[31:00] and C/BE[3:0] and the PAR lines. During the data phase of all Memory Write commands, the PCnet-PCI controller checks the PERR input to detect whether the target has reported a parity error.

DATAPERR is set by the PCnet-PCI controller and cleared by writing a ONE. Writing a ZERO has no effect. DATAPERR is not affected by H_RESET or S_RESET or asserting the SLEEP pin.

7–0 RES — Reserved locations. Read as ZERO, write operations have no effect.

Revision ID Register (Offset 08h)

The Revision ID register is an 8-bit register that specifies the PCnet-PCI controller revision number. The current value of this register is 00h.

The Revision ID register is located at offset 08h in the PCI Configuration Space. It is read only.

Programming Interface Register (Offset 09h)

The Programming Interface register is an 8-bit register that identifies the programming interface of PCnet-PCI controller. PCI does not define any specific register-level programming interfaces for network devices. The value of this register is 00h.

The Programming Interface register is located at address 09h in the PCI Configuration Space. It is read only.

Sub-Class Register (Offset 0Ah)

The Sub-Class register is an 8-bit register that identifies specifically the function of the PCnet-PCI controller. The value of this register is 00h which identifies the PCnet-PCI device as an Ethernet controller.

The Sub-Class register is located at offset 0Ah in the PCI Configuration Space. It is read only.

Base-Class Register (Offset 0Bh)

The Base-Class register is an 8-bit register that broadly classifies the function of the PCnet-PCI controller. The value of this register is 02h which classifies the PCnet-PCI device as a network controller.

The Base-Class register is located at offset 0Bh in the PCI Configuration Space. It is read only.

Latency Timer Register (Offset 0Dh)

The Latency Timer register is an 8-bit register that specifies the maximum time the PCnet-PCI controller can continue with bus master transfers after the system arbiter has removed $\overline{GNT}$. The time is measured in CLK cycles. The working copy of the timer will start counting down when the PCnet-PCI controller asserts $\overline{FRAME}$ for the first time during a bus mastership period. The counter will freeze at ZERO. When the counter is ZERO and $\overline{GNT}$ is deasserted by the system arbiter, the PCnet-PCI controller will finish the current data phase and then immediately release the bus.

The value for the PCnet-PCI controller Latency Timer register is 00h, which indicates that, when the PCnet-PCI controller is preempted, it will always release the bus immediately after finishing the current data phase.

The Latency Timer register is located at offset 0Dh in the PCI Configuration Space. It is read only.

Header Type Register (Offset 0Eh)

The Header Type register is an 8-bit register that describes the format of the PCI Configuration Space locations 10h to 3Ch and that identifies a device to be single or multi function. The Header Type register is located at offset 0Eh in the PCI Configuration Space. It is read only.

7	FUNCT	Single function/multi function device. Read as ZERO, write operations have no effect. The PCnet-PCI controller is a single function device.
6–0	LAYOUT	PCI configuration space layout. Read as ZERO, write operations have no effect. The layout of the PCI configuration space locations 10h to 3Ch is as show in the table at the beginning of this section.

Base Address Register (Offset 10h)

The Base Address register is a 32-bit register that determines the location of the PCnet-PCI controller in all of I/O space. It is located at offset 10h in the PCI Configuration Space.

31–5	IOBASE	I/O base address significant 27 bits. These bits are written by the host to specify the location of the PCnet-PCI controller in all of I/O space. IOBASE must be written with a valid address before the PCnet-PCI controller slave I/O mode is turned on with setting the IOEN bit (bit 0 in the Command register). When the PCnet-PCI controller is enabled for I/O mode (IOEN is set), it monitors the PCI bus for a valid I/O command. If the value on AD[31:05] during the address phase of the cycles matches the value of IOBASE, the PCnet-PCI controller will drive $\overline{DEVSEL}$ indicating it will respond to the access. IOBASE is read and written by the host. IOBASE is not effected by H_RESET or S_RESET or asserting the $\overline{SLEEP}$ pin.
4–2	IOSIZE	I/O size requirements. Read as ZERO, write operations have no effect. IOSIZE indicates the size of the I/O space the PCnet-PCI controller requires. When the host writes a value of FFFF FFFFh to the Base Address register, it will read back a value of “0” in bits 4–2. That indicates a PCnet-PCI I/O space requirement of 32 bytes.

1	RES	Reserved location. Read as ZERO, write operations have no effect.
0	IOSPACE	I/O space indicator. Read as ONE, write operations have no effect. Indicating that this Base Address register describes an I/O base address.

Interrupt Line Register (Offset 3Ch)

The Interrupt Line register is an 8-bit register that is used to communicate the routing of the interrupt. This register is written by the POST software as it initialized the PCnet-PCI controller in the system. The register is read by the network driver to determine the interrupt channel which the POST software has assigned to the PCnet-PCI controller. The Interrupt Line register is not modified by the PCnet-PCI controller. It has no effect on the operation of the device.

The Interrupt Line register is located at offset 3Ch in the PCI Configuration Space. It is read an written by the host. It is not effected by H_RESET or S_RESET or asserting the $\overline{\text{SLEEP}}$ pin.

Interrupt Pin Register (Offset 3Dh)

This Interrupt Pin register is an 8-bit register indicating the interrupt pin the PCnet-PCI controller is using. The value for the PCnet-PCI Interrupt Pin register is 01h, which corresponds to INTA.

The Interrupt Pin register is located at offset 3Dh in the PCI Configuration Space. It is read only.

RAP Register

The RAP (Register Address Pointer) register is used to gain access to CSR and BCR registers on board the PCnet-PCI controller. The value of the RAP indicates the address of a CSR or BCR whenever an RDP or BDP access is performed. That is to say, RAP serves as a pointer to CSR and BDP space.

As an example of RAP use, consider a read access to CSR4. In order to access this register, it is necessary to first load the value 0004h into the RAP by performing a write access to the RAP offset of 12h (12h when WIO mode has been selected, 14h when DWIO mode has been selected). Then a second access is performed PCnet-PCI controller, this time to the RDP offset of 10h (for either WIO or DWIO mode). The RDP access is a read access, and since RAP has just been loaded with the value of 0004h, the RDP read will yield the contents of CSR4. A read of the BDP at this time (offset of 16h when WIO mode has been selected, 1Ch when DWIO mode has been selected) will yield the contents of BCR4, since the RAP is used as the pointer into both BDP and RDP space.

RAP: Register Address Port

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–8	RES	Reserved locations. Read and written as ZEROs.
7–0	RAP	Register Address Port. The value of these 8 bits determines which CSR or BCR will be accessed when an I/O access to the RDP or BDP port, respectively, is performed. A write access to undefined CSR or BCR locations may cause unexpected reprogramming of the PCnet-PCI control registers. A read access will yield undefined values. RAP is cleared by H_RESET or S_RESET and is unaffected by the STOP bit.

Control and Status Registers

The CSR space is accessible by performing accesses to the RDP (Register Data Port). The particular CSR that is read or written during an RDP access will depend upon the current setting of the RAP. RAP serves as a pointer into the CSR space. RAP also serves as the pointer to BCR space, which is described in a later section.

CSR0: PCnet-PCI Controller Status Register

Bit	Name	Description
		Certain bits in CSR0 indicate the cause of an interrupt. The register is designed so that these indicator bits are cleared by writing ONEs to those bit locations. This means that the software can read CSR0 and write back the value just read to clear the interrupt condition.
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15	ERR	Error is set by the ORing of BABL, CERR, MISS, and MERR. ERR remains set as long as any of the error flags are true. ERR is read only. Write operations are ignored.
14	BABL	Babble is a transmitter time-out error. It indicates that the transmitter has been on the channel

longer than the time required to send the maximum length frame. BABL will be set if 1519 bytes or greater are transmitted.

When BABL is set, $\overline{INTA}$ is asserted if IENA = 1 and the mask bit BABLM in CSR3 is clear. BABL assertion will set the ERR bit.

BABL is set by the MAC layer and cleared by writing a "1". Writing a "0" has no effect. BABL is cleared by H_RESET or S_RESET or by setting the STOP bit.

13 CERR — Collision Error indicates that the collision inputs to the AUI port failed to activate within 20 network bit times after the chip terminated transmission (SQE Test). This feature is a transceiver test feature.

In 10BASE-T mode, CERR will be set after a transmission if the T-MAU is in link fail state.

CERR assertion will not result in an interrupt being generated. CERR assertion will set the ERR bit.

CERR is set by the MAC layer and cleared by writing a "1". Writing a "0" has no effect. CERR is cleared by H_RESET or S_RESET or by setting the STOP bit.

12 MISS — Missed Frame is set when PCnet-PCI controller has lost an incoming receive frame resulting from a Receive Descriptor not being available. This bit is the only immediate indication that receive data has been lost since there is no current receive descriptor. Missed Frame Counter (CSR112) also increments each time a receive frame is missed.

When MISS is set, $\overline{INTA}$ is asserted if IENA = 1 and the mask bit MISSM in CSR3 is clear. MISS assertion will set the ERR bit.

MISS is set by the Buffer Management Unit and cleared by writing a "1". Writing a "0" has no effect. MISS is cleared by H_RESET or S_RESET or by setting the STOP bit.

11 MERR — Memory Error is set when PCnet-PCI controller requests the use of the system interface bus by asserting $\overline{REQ}$ and has not received $\overline{GNT}$ assertion after a programmable length of time. The length of time in microseconds before MERR is asserted will depend upon the setting of the Bus Timeout Register (CSR100). The default setting of CSR100 will give a MERR after 51.2 microseconds of bus latency.

When MERR is set, $\overline{INTA}$ is asserted if IENA = 1 and the mask bit MERRM in CSR3 is clear. MERR assertion will set the ERR bit, regardless of the settings of IENA and MERRM.

MERR is set by the Bus Interface Unit and cleared by writing a "1". Writing a "0" has no effect. MERR is cleared by H_RESET, S_RESET or by setting the STOP bit.

10 RINT — Receive Interrupt. RINT is set by the Buffer Management Unit of the PCnet-PCI controller after the last descriptor of a receive packet has been updated by writing a ZERO to the ownership bit. RINT may also be set when the first descriptor of a receive packet has been updated by writing a ZERO to the ownership bit if the SPRINTEN bit of CSR3 has been set to a ONE.

When RINT is set, $\overline{INTA}$ is asserted if IENA = 1 and the mask bit RINTM in CSR3 is clear.

RINT is cleared by the host by writing a "1". Writing a "0" has no effect. RINT is cleared by H_RESET, S_RESET or by setting the STOP bit.

9 TINT — Transmit Interrupt is set after the OWN bit in the last descriptor of a transmit frame has been cleared to indicate the frame has been sent or an error occurred in the transmission.

When TINT is set, $\overline{INTA}$ is asserted if IENA = 1 and the mask bit TINTM in CSR3 is clear.

TINT is set by the Buffer Management Unit and cleared by writing a "1". Writing a "0" has no effect. TINT is cleared by H_RESET or S_RESET or by setting the STOP bit.

8 IDON — Initialization Done indicates that the initialization sequence has

		completed. When IDON is set, PCnet-PCI controller has read the Initialization block from memory. When IDON is set, $\overline{\text{INTA}}$ is asserted if IENA = 1 and the mask bit IDONM in CSR3 is clear. IDON is set by the Buffer Management Unit after the initialization block has been read from memory and cleared by writing a "1". Writing a "0" has no effect. IDON is cleared by H_RESET or S_RESET or by setting the STOP bit.
7	INTR	Interrupt Flag indicates that one or more following interrupt causing conditions has occurred: BABL, MISS, MERR, MPCO, RCVCCO, RINT, RPCO, TINT, IDON, JAB or TXSTRT; and its associated mask bit is clear. If IENA = 1 and INTR is set, $\overline{\text{INTA}}$ will be active. INTR is read only. INTR is cleared by H_RESET, S_RESET, setting the STOP bit or by clearing all of the active individual interrupt bits that have not been masked out.
6	IENA	Interrupt Enable allows $\overline{\text{INTA}}$ to be active if the Interrupt Flag is set. If IENA = 0 then $\overline{\text{INTA}}$ will be disabled regardless of the state of INTR. IENA is set by writing a "1" and cleared by writing a "0". IENA is cleared by H_RESET or S_RESET or by setting the STOP bit.
5	RXON	Receive On indicates that the Receive function is enabled. RXON is set if DRX = 0 in CSR15[1] after the START bit is set. If INIT and START are set together, RXON will not be set until after the initialization block has been read in. RXON is read only. RXON is cleared by H_RESET or S_RESET or by setting the STOP bit.
4	TXON	Transmit On indicates that the Transmit function is enabled. TXON is set if DTX = 0 in CSR15[1] after the START bit is set. If INIT and START are set together, TXON will not be set until after the initialization block has been read in. TXON is read only. TXON is cleared by H_RESET or S_RESET or by setting the STOP bit.
3	TDMD	Transmit Demand, when set, causes the Buffer Management Unit to access the Transmit Descriptor Ring without waiting for the poll-time counter to elapse. If TXON is not enabled, TDMD bit will be reset and no Transmit Descriptor Ring access will occur. TDMD is required to be set if the DPOLL bit in CSR4 is set; setting TDMD while DPOLL = 0 merely hastens the PCnet-PCI controller's response to a Transmit Descriptor Ring Entry. TDMD is set by writing a "1". Writing a "0" has no effect. TDMD will be cleared by the Buffer Management Unit when it fetches a Transmit Descriptor. TDMD is cleared by H_RESET or S_RESET and setting the STOP bit.
2	STOP	STOP assertion disables the chip from all DMA activity. The chip remains inactive until either STRT or INIT are set. If STOP, STRT and INIT are all set together, STOP will override STRT and INIT. STOP is set by writing a "1", by H_RESET or S_RESET. Writing a "0" has no effect. STOP is cleared by setting either STRT or INIT.
1	STRT	STRT assertion enables PCnet-PCI controller to send and receive frames, and perform buffer management operations. Setting STRT clears the STOP bit. If STRT and INIT are set together, PCnet-PCI controller initialization will be performed first. STRT is set by writing a "1". Writing a "0" has no effect. STRT is cleared by H_RESET, S_RESET or by setting the STOP bit.
0	INIT	INIT assertion enables PCnet-PCI controller to begin the initialization procedure which reads in the initialization block from memory. Setting INIT clears the STOP bit. If STRT and INIT are set together, PCnet-PCI controller initialization will be performed first. INIT is not cleared when the

initialization sequence has completed.

INIT is set by writing a "1". Writing a "0" has no effect. INIT is cleared by H_RESET, S_RESET or by setting the STOP bit.

CSR1: IADR[15:0]

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	IADR[15:0]	Lower 16 bits of the address of the Initialization Block. Bit locations 1 and 0 must both be ZERO to align the initialization block to a double-word boundary, regardless of the value of SSIZE32 (BCR20/CSR58, bit 8). This register is aliased with CSR16. Read/Write accessible only when the STOP bit in CSR0 is set. Unaffected by H_RESET or S_RESET or by setting the STOP bit.

CSR2: IADR[31:16]

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–8	IADR[31:24]	If SSIZE32 is set (BCR20, bit 8), then the IADR[31:24] bits will be used strictly as the upper 8 bits of the initialization block address. However, if SSIZE32 is reset (BCR20, bit 8), then the IADR[31:24] bits will be used to generate the upper 8 bits of all bus mastering addresses, as required for a 32 bit address bus. Note that the 16-bit software structures specified by the SSIZE32=0 setting will yield only 24 bits of address for PCnet-PCI bus master accesses, while the 32-bit hardware for which the PCnet-PCI controller is intended will require 32 bits of address. Therefore, whenever SSIZE32=0, the IADR[31:24] bits will be appended to the 24-bit initialization address, to each 24-bit descriptor base address and to each beginning 24-bit buffer address in order to form complete 32-bit addresses. The upper 8 bits that exist in the descriptor address registers and the buffer address registers which are stored on board the PCnet-PCI controller will be overwritten with the IADR[31:24] value, so that CSR accesses to these registers will show the 32 bit address that includes the appended field. If SSIZE32=1, then software will provide 32-bit pointer values for all of the shared software structures – i.e. descriptor bases and buffer addresses, and therefore, IADR[31:24] will not be written to the upper 8 bits of any of these resources, but it will be used as the upper 8 bits of the initialization address. This register is aliased with CSR17. Read/Write accessible only when the STOP bit in CSR0 is set. Unaffected by H_RESET, S_RESET or by setting the STOP bit.
7–0	IADR[23:16]	Bits 23 through 16 of the address of the Initialization Block. Whenever this register is written, CSR17 is updated with CSR2's contents. Read/Write accessible only when the STOP bit in CSR0 is set. Unaffected by H_RESET, S_RESET or by setting the STOP bit.

CSR3: Interrupt Masks and Deferral Control

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15	RES	Reserved location. Read and written as ZERO.
14	BABLM	Babble Mask. If BABLM is set, the BABL bit in CSR0 will be masked and unable to set INTR flag in CSR0. Read/Write accessible always. BABLM is cleared by H_RESET or S_RESET and is not affected by STOP.
13	RES	Reserved location. Read and written as ZERO.
12	MISSM	Missed Frame Mask. If MISSM is set, the MISS bit in CSR0 will be masked and unable to set INTR flag in CSR0. Read/Write accessible always. MISSM is cleared by H_RESET

		or S_RESET and is not affected by STOP.
11	MERRM	Memory Error Mask. If MERRM is set, the MERR bit in CSR0 will be masked and unable to set INTR flag in CSR0. Read/Write accessible always. MERRM is cleared by H_RESET or S_RESET and is not affected by STOP.
10	RINTM	Receive Interrupt Mask. If RINTM is set, the RINT bit in CSR0 will be masked and unable to set INTR flag in CSR0. Read/Write accessible always. RINTM is cleared by H_RESET or S_RESET and is not affected by STOP.
9	TINTM	Transmit Interrupt Mask. If TINTM is set, the TINT bit in CSR0 will be masked and unable to set INTR flag in CSR0. Read/Write accessible always. TINTM is cleared by H_RESET or S_RESET and is not affected by STOP.
8	IDONM	Initialization Done Mask. If IDONM is set, the IDON bit in CSR0 will be masked and unable to set INTR flag in CSR0. Read/Write accessible always. IDONM is cleared by H_RESET or S_RESET and is not affected by STOP.
7–6	RES	Reserved locations. Read and written as ZEROs.
5	LAPPEN	Look-Ahead Packet Processing Enable. When set to a ONE, the LAPPEN bit will cause the PCnet-PCI controller to generate an interrupt following the descriptor write operation to the first buffer of a receive packet. This interrupt will be generated in addition to the interrupt that is generated following the descriptor write operation to the last buffer of a receive packet. The interrupt will be signaled through the RINT bit of CSR0. Setting LAPPEN to a ONE also enables the PCnet-PCI controller to read the STP bit of receive descriptors. PCnet-PCI controller will use the STP information to determine where it should begin writing a receive packets data. Note that while in this mode, the PCnet-PCI controller can write intermediate packet data to buffers whose descriptors do not contain STP bits set to ONE. Following the write to the last descriptor used by a packet, the PCnet-PCI controller will scan through the next descriptor entries to locate the next STP bit that is set to a ONE. The PCnet-PCI controller will begin writing the next packets data to the buffer pointed to by that descriptor. Note that because several descriptors may be allocated by the host for each packet, and not all messages may need all of the descriptors that are allocated between descriptors that contain STP = ONE, then some descriptors/buffers may be skipped in the ring. While performing the search for the next STP bit that is set to ONE, the PCnet-PCI controller will advance through the receive descriptor ring regardless of the state of ownership bits. If any of the entries that are examined during this search indicate PCnet-PCI controller ownership of the descriptor but also indicate STP = 0, then the PCnet-PCI controller will reset the OWN bit to ZERO in these entries. If a scanned entry indicates host ownership with STP = 0, then the PCnet-PCI controller will not alter the entry, but will advance to the next entry. When the STP bit is found to be true, but the descriptor that contains this setting is not owned by the PCnet-PCI controller, then the PCnet-PCI controller will stop advancing through the ring entries and begin periodic polling of this entry. When the STP bit is found to be true, and the descriptor that contains this setting is owned by the PCnet-PCI controller, then the PCnet-PCI controller will stop advancing through the ring entries, store the descriptor information that it has just read, and wait for the next receive to arrive. This behavior allows the host software to pre-assign buffer space in such a manner that the “header” portion of a receive packet will always be written to a particular memory area, and the “data” portion of a receive packet

		will always be written to a separate memory area. The interrupt is generated when the "header" bytes have been written to the "header" memory area. Read/Write accessible always. The LAPPEN bit will be reset to ZERO by H_RESET or S_RESET and will be unaffected by STOP. See Appendix D for more information on the LAPP concept.
4	DXMT2PD	Disable Transmit Two Part Deferral (see Medium Allocation section in Media Access Management fro more details). If DXMT2PD is set, Transmit Two Part Deferral will be disabled. Read/Write accessible always. DXMT2PD is cleared by H_RESET or S_RESET and is not affected by STOP.
3	EMBA	Enable Modified Back-off Algorithm (see Contention Resolution section in Media Access Management for more details). If EMBA is set, a modified back-off algorithm is implemented. Read/Write accessible always. EMBA is cleared by H_RESET or S_RESET and is not affected by STOP.
2	BSWP	Byte Swap. This bit is used to choose between big and little Endian modes of operation. When BSWP is set to a ONE, big Endian mode is selected. When BSWP is set to ZERO, little Endian mode is selected. When big Endian mode is selected, the PCnet-PCI controller will swap the order of bytes on the AD bus during a data phase on accesses to the FIFOs only. Specifically, AD[31:24] becomes BYTE0, AD[23:16] becomes BYTE1, AD[15:8] becomes BYTE2 and AD[7:0] becomes BYTE3 when big Endian mode is selected. When little Endian mode is selected, the order of bytes on the AD bus during a data phase is: AD[31:24] is BYTE3, AD[23:16] is BYTE2, AD[15:8] is BYTE1 and AD[7:0] is BYTE0. Byte swap only affects data transfers that involve the FIFOs. Initialization block transfers are not affected by the setting of the BSWP bit. Descriptor transfers are not affected by the setting of the BSWP bit. RDP, RAP and BDP accesses are not affected by the setting of the BSWP bit. APROM transfers are not affected by the setting of the BSWP bit. Note that the byte ordering of the PCI bus is defined to be little endian. BSWP must not be set to ONE when the PCnet-PCI controller operates in a PCI system. BSWP is write/readable regardless of the state of the STOP bit. BSWP is cleared by H_RESET or S_RESET and is not affected by STOP bit.
1	RES	Reserved location. The default value of this bit is a ZERO. Writing a ONE to this bit has no effect on device function; If a ONE is written to this bit, then a ONE will be read back. Existing drivers may write a ONE to this bit for compatibility, but new drivers should write a ZERO to this bit and should treat the read value as undefined.
0	RES	Reserved location. The default value of this bit is a ZERO. Writing a ONE to this bit has no effect on device function. If a ONE is written to this bit, then a ONE will be read back. Existing drivers may write a ONE to this bit for compatibility, but new drivers should write a ZERO to this bit and should treat the read value as undefined.

CSR4: Test and Features Control

Bit	Name	Description
		Certain bits in CSR4 indicate the cause of an interrupt. The register is designed so that these indicator bits are cleared by writing ONEs to those bit locations. This means that the software can read CSR4 and write back the value just read to clear the interrupt condition.
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15	ENTST	Enable Test Mode operation. Setting ENTST to ONE enables internal test functions which are useful only for stand alone integrated circuit testing. In addition,

		the Runt Packet Accept (RPA) bit (CSR124, bit 3) may be changed only when ENTST is set to ONE. To enable RPA, the user must first write a ONE to the ENTST bit. Next, the user must first write a ONE to the RPA bit (CSR124, bit 3). Finally, the user must write a ZERO to the ENTST bit to take the device out of test mode operation. Once, the RPA bit has been set to ONE, the device will remain in the Runt Packet Accept mode until the RPA bit is cleared to ZERO. Read/Write accessible. ENTST is cleared by H_RESET or S_RESET and is unaffected by the STOP bit.
14	DMAPLUS	When DMAPLUS = "1", DMA Burst Counter in CSR80 is disabled. If DMAPLUS = "0", the counter is enabled. Read/Write accessible. DMAPLUS is cleared by H_RESET or S_RESET and is unaffected by the STOP bit.
13	TIMER	Timer Enable Register. If TIMER is set, the Bus Timer Register, CSR82 is enabled. If TIMER is cleared, the Bus Timer Register is disabled. Read/Write accessible. TIMER is cleared by H_RESET or S_RESET and is unaffected by the STOP bit.
12	DPOLL	Disable Transmit Polling. If DPOLL is set, the Buffer Management Unit will disable transmit polling. Likewise, if DPOLL is cleared, automatic transmit polling is enabled. If DPOLL is set, TDMD bit in CSR0 must be set in order to initiate a manual poll of a transmit descriptor. Transmit descriptor polling will not take place if TXON is reset. Read/Write accessible. DPOLL is cleared by H_RESET or S_RESET and is unaffected by the STOP bit.
11	APAD_XMT	Auto Pad Transmit. When set, APAD_XMT enables the automatic padding feature. Transmit frames will be padded to extend them to 64 bytes including FCS. The FCS is calculated for the entire frame including pad, and appended after the pad field. APAD_XMT will override the programming of the DXMTFCS bit. Read/Write accessible. APAD_XMT is cleared by H_RESET or S_RESET and is unaffected by the STOP bit.
10	ASTRP_RCV	Auto Strip Receive. When set, ASTRP_RCV enables the automatic pad stripping feature. The pad and FCS fields will be stripped from receive frames and not placed in the FIFO. Read/Write accessible. ASTRP_RCV is cleared by H_RESET or S_RESET and is unaffected by the STOP bit.
9	MFCO	Missed Frame Counter Overflow Interrupt. Indicates the MFC (CSR112) wrapped around. Can be cleared by writing a 1 to this bit. Also cleared by H_RESET, S_RESET or by asserting the STOP bit. Writing a 0 has no effect. When MFCO is set, $\overline{\text{INTA}}$ is asserted if IENA is ONE and the mask bit MFCOM is ZERO. When the value 01h has been programmed into the SWSTYLE register (BCR20, bits 7–0) for ILACC (Am79C900) compatibility, then this bit has no meaning and PCnet-PCI controller will never set the value of this bit to ONE.
8	MFCOM	Missed Frame Counter Overflow Mask. If MFCOM is set, MFCO will be unable to set INTR in CSR0. Set to a ONE by H_RESET or S_RESET, unaffected by the STOP bit. When the value 01h has been programmed into the SWSTYLE register (BCR20, bits 7–0) for ILACC (Am79C900) compatibility, then this bit has no meaning and PCnet-PCI controller will set the value of this bit to ZERO.
7	RES	Reserved location. Written as ZERO and read as ZERO.
6	RES	Reserved location. This bit may be written to as either a ONE or a ZERO, but will always be read as a ZERO. This bit has no effect on PCnet-PCI controller operation.
5	RCVCCO	Receive Collision Counter Overflow. Indicates the Receive Collision Counter (CSR114) wrapped around. Can be cleared by

writing a 1 to this bit. Also cleared by H_RESET, S_RESET or by setting the STOP bit. Writing a 0 has no effect.

When RCVCCO is set, $\overline{INTA}$ is asserted if IENA is ONE and the mask bit RCVCCOM is ZERO.

When the value 01h has been programmed into the SWSTYLE register (BCR20, bits 7–0) for ILACC (Am79C900) compatibility, then this bit has no meaning and PCnet-PCI controller will never set the value of this bit to ONE.

4 RCVCCOM — Receive Collision Counter Overflow Mask.

If RCVCCOM is set, RCVCCO will be unable to set INTR in CSR0.

RCVCCOM is set by H_RESET or S_RESET and is not affected by STOP bit.

When the value 01h has been programmed into the SWSTYLE register (BCR20, bits 7–0) for ILACC (Am79C900) compatibility, then this bit has no meaning and PCnet-PCI controller will set the value of this bit to ZERO.

3 TXSTRT — Transmit Start status is set whenever PCnet-PCI controller begins transmission of a frame.

When TXSTRT is set, $\overline{INTA}$ is asserted if IENA = 1 and the mask bit TXSTRTM (CSR4 bit 2) is clear.

TXSTRT is set by the MAC Unit and cleared by writing a “1”, by H_RESET, S_RESET or by asserting the STOP bit. Writing a “0” has no effect.

2 TXSTRTM — Transmit Start Mask. If TXSTRTM is set, the TXSTRT bit in CSR4 will be masked and unable to set INTR flag in CSR0.

Read/Write accessible. TXSTRTM is set by H_RESET or S_RESET and is not affected by the STOP bit.

1 JAB — Jabber Error is set when the PCnet-PCI controller Twisted-pair MAU function exceeds an allowed transmission limit. Jabber is set by the TMAU cell and can only be asserted in10BASE-T mode.

When JAB is set, $\overline{INTA}$ is asserted if IENA = 1 and the mask bit JABM (CSR4 bit 0) is clear.

JAB is set by the TMAU circuit and cleared by writing a “1”. Writing a “0” has no effect. JAB is also cleared by H_RESET, S_RESET or by asserting the STOP bit.

When the value 01h has been programmed into the SWSTYLE register (BCR20, bits 7–0) for ILACC (Am79C900) compatibility, then this bit has no meaning and PCnet-PCI controller will never set the value of this bit to ONE.

0 JABM — Jabber Error Mask. If JABM is set, the JAB bit in CSR4 will be masked and unable to set INTR flag in CSR0.

Read/Write accessible. JABM is set by H_RESET or S_RESET and is not affected by the STOP bit.

When the value 01h has been programmed into the SWSTYLE register (BCR20, bits 7–0) for ILACC (Am79C900) compatibility, then this bit has no meaning and PCnet-PCI controller will set the value of this bit to ZERO.

CSR6: RX/TX Descriptor Table Length

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–12	TLEN	Contains a copy of the transmit encoded ring length (TLEN) field read from the initialization block during PCnet-PCI controller initialization. This field is written during the PCnet-PCI controller initialization routine. Read accessible only when STOP bit is set. Write operations have no effect and should not be performed. TLEN is only defined after initialization. These bits are unaffected by H_RESET, S_RESET or STOP.
11–8	RLEN	Contains a copy of the receive encoded ring length (RLEN) read from the initialization block during PCnet-PCI controller initialization. This field is written during

		the PCnet-PCI controller initialization routine. Read accessible only when STOP bit is set. Write operations have no effect and should not be performed. RLEN is only defined after initialization. These bits are unaffected by H_RESET, S_RESET or STOP.
7–0	RES	Reserved locations. Read as ZERO. Write operations should not be performed.

CSR8: Logical Address Filter, LADRF[15:0]

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	LADRF[15:0]	Logical Address Filter, LADRF[15:0]. The content of this register is undefined until loaded from the initialization block after the INIT bit in CSR0 has been set or a direct I/O write has been performed on this register. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR9: Logical Address Filter LADRF[31:16]

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	LADRF[31:16]	Logical Address Filter, LADRF[31:16]. The content of this register is undefined until loaded from the initialization block after the INIT bit in CSR0 has been set or a direct I/O write has been performed on this register. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR10: Logical Address Filter, LADRF[47:32]

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	LADRF[47:32]	Logical Address Filter, LADRF[47:32]. The content of this register is undefined until loaded from the initialization block after the INIT bit in CSR0 has been set or a direct I/O write has been performed on this register. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR11: Logical Address Filter, LADRF[63:48]

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	LADRF[63:48]	Logical Address Filter, LADRF[63:48]. The content of this register is undefined until loaded from the initialization block after the INIT bit in CSR0 has been set or a direct I/O write has been performed on this register. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR12: Physical Address Register, PADR[15:0]

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	PADR[15:0]	Physical Address Register, PADR[15:0]. The content of this register is undefined until loaded from the initialization block after the INIT bit in CSR0 has been set or a direct I/O write has been performed on this register. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR13: Physical Address Register, PADR[31:16]

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	PADR[31:16]	Physical Address Register, PADR[31:16]. The content of this register is undefined until loaded from the initialization block after the INIT bit in CSR0 has been set or a direct I/O write has been performed on this register. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR14: Physical Address Register, PADR[47:32]

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	PADR[47:32]	Physical Address Register, PADR[47:32]. The content of this register is undefined until loaded from the initialization block after the INIT bit in CSR0 has been set or a direct I/O write has been performed on this register. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR15: Mode Register

Bit	Name	Description
		This register's fields are loaded during the PCnet-PCI controller initialization routine with the corresponding Initialization Block values or a direct I/O write has been performed on this register.
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15	PROM	Promiscuous Mode. When PROM = "1", all incoming receive frames are accepted. Read/write accessible only when STOP bit is set.
14	DRCVBC	Disable Receive Broadcast. When set, disables the PCnet-PCI controller from receiving broadcast messages. Used for protocols that do not support broadcast addressing, except as a function of multicast. DRCVBC is cleared by activation of H_RESET or S_RESET (broadcast messages will be received) and is unaffected by STOP. Read/write accessible only when STOP bit is set.
13	DRCVPA	Disable Receive Physical Address. When set, the physical address detection (Station or node ID) of the PCnet-PCI controller will be disabled. Frames addressed to the nodes individual physical address will not be recognized. Read/write accessible only when STOP bit is set.
12	DLNKTST	Disable Link Status. When DLNKTST = "1", monitoring of Link Pulses is disabled. When DLNKTST = "0", monitoring of Link Pulses is enabled. This pin only has meaning when the 10BASE-T network interface is selected. Read/write accessible only when STOP bit is set.
11	DAPC	Disable Automatic Polarity Correction. When DAPC = "1", the 10BASE-T receive polarity reversal algorithm is disabled. Likewise, when DAPC = "0", the polarity reversal algorithm is enabled. This bit only has meaning when the 10BASE-T network interface is selected. Read/write accessible only when STOP bit is set.
10	MENDECL	MENDEC Loopback Mode. See the description of the LOOP bit in CSR15. Read/write accessible only when STOP bit is set.
9	LRT	Low Receive Threshold (T-MAU Mode only)
	TSEL	Transmit Mode Select (AUI Mode only)
	LRT	Low Receive Threshold. When LRT = "1", the internal twisted pair receive thresholds are reduced by 4.5 dB below the standard 10BASE-T value (approximately 3/5) and the unsquelch threshold for the RXD circuit will be 180 mV – 312 mV peak. When LRT = "0", the unsquelch threshold for the RXD circuit will be the standard 10BASE-T value, 300 – 520 mV peak. In either case, the RXD circuit post squelch threshold will be one half of the unsquelch threshold. This bit only has meaning when the 10BASE-T network interface is selected. Read/write accessible only when STOP bit is set. Cleared by H_RESET or S_RESET and is unaffected by STOP.
	TSEL	Transmit Mode Select. TSEL controls the levels at which the

AUI drivers rest when the AUI transmit port is idle. When TSEL = 0, DO+ and DO– yield "zero" differential to operate transformer coupled loads (Ethernet 2 and 802.3). When TSEL = 1, the DO+ idles at a higher value with respect to DO– , yielding a logical HIGH state (Ethernet 1).

This bit only has meaning when the AUI network interface is selected.

Read/write accessible only when STOP bit is set. Cleared by H_RESET or S_RESET.

8–7 PORTSEL[1:0] Port Select bits allow for software controlled selection of the network medium.

PORTSEL settings of AUI and 10BASE-T are ignored when the ASEL bit of BCR2 (bit 1) has been set to ONE.

The network port configurations are as follows:

PORTSEL CSR15[1:0]	ASEL (BCR2 [1])	Link Status (of 10BASE-T)	Network Port
0 X	1	Fail	AUI
0 X	1	Pass	10BASE-T
0 0	0	X	AUI
0 1	0	X	10BASE-T
1 0	X	X	Reserved
1 1	X	X	Reserved

Read/write accessible only when STOP bit is set. Cleared by H_RESET or S_RESET and is unaffected by STOP.

6 INTL Internal Loopback. See the description of LOOP, CSR15[2].

Read/write accessible only when STOP bit is set.

5 DRTY Disable Retry. When DRTY = "1", PCnet-PCI controller will attempt only one transmission. If DRTY = "0", PCnet-PCI controller will attempt 16 transmissions before signaling a retry error.

Read/write accessible only when STOP bit is set.

4 FCOLL Force Collision. This bit allows the collision logic to be tested. PCnet-PCI controller must be in internal loopback for FCOLL to be valid. If FCOLL = "1", a collision will be forced during loopback transmission attempts; a Retry Error will ultimately result. If FCOLL = "0", the Force Collision logic will be disabled. FCOLL is defined after the Initialization Block is read.

Read/write accessible only when STOP bit is set.

3 DXMTFCS Disable Transmit CRC (FCS). When DXMTFCS = 0, the transmitter will generate and append a FCS to the transmitted frame. When DXMTFCS = 1, the FCS logic is allocated to the receiver and no FCS is generated or sent with the transmitted frame. DXMTFCS is overridden when ADD_FCS is set in TMD1.

See also the ADD_FCS bit in TMD1. If DXMTFCS is set and ADD_FCS is clear for a particular frame, no FCS will be generated. The value of ADD_FCS is valid only when STP is set in TMD1. If ADD_FCS is set for a particular frame, the state of DXMTFCS is ignored and a FCS will be appended on that frame by the transmit circuitry.

In loopback mode, this bit determines if the transmitter appends FCS or if the receiver checks the FCS.

This bit was called DTCR in the LANCE (Am7990).

Read/write accessible only when STOP bit is set.

2 LOOP Loopback Enable allows PCnet-PCI controller to operate in full duplex mode for test purposes. When LOOP = "1", loopback is enabled. In combination with INTL and MENDECL, various loopback modes are defined as follows:

LOOP	INTL	MENDECL	Loopback Mode
0	X	X	Non-loopback
1	0	X	External Loopback
1	1	0	Internal Loopback Include MENDEC
1	1	1	Internal Loopback Exclude MENDEC

Read/write accessible only when STOP bit is set. LOOP is cleared by H_RESET or S_RESET and is unaffected by STOP.

1 DTX Disable Transmit results in PCnet-PCI controller not accessing the Transmit Descriptor Ring

		and therefore no transmissions are attempted. DTX = "0", will set TXON bit (CSR0 bit 4) if STRT (CSR0 bit 1) is asserted. Read/write accessible only when STOP bit is set.
0	DRX	Disable Receiver results in PCnet-PCI controller not accessing the Receive Descriptor Ring and therefore all receive frame data are ignored. DRX = "0", will set RXON bit (CSR0 bit 5) if STRT (CSR0 bit 1) is asserted. Read/write accessible only when STOP bit is set.

CSR16: Initialization Block Address Lower

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	IADR[15:0]	This register is an alias of CSR1.

CSR17: Initialization Block Address Upper

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	IADR[31:16]	This register is an alias of CSR2.

CSR18: Current Receive Buffer Address Lower

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	CRBAL	Contains the lower 16 bits of the current receive buffer address at which the PCnet-PCI controller will store incoming frame data. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR19: Current Receive Buffer Address Upper

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	CRBAU	Contains the upper 16 bits of the current receive buffer address at which the PCnet-PCI controller will store incoming frame data. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR20: Current Transmit Buffer Address Lower

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	CXBAL	Contains the lower 16 bits of the current transmit buffer address from which the PCnet-PCI controller is transmitting. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR21: Current Transmit Buffer Address Upper

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	CXBAU	Contains the upper 16 bits of the current transmit buffer address from which the PCnet-PCI controller is transmitting. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR22: Next Receive Buffer Address Lower

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	NRBAL	Contains the lower 16 bits of the next receive buffer address to which the PCnet-PCI controller will store incoming frame data. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR23: Next Receive Buffer Address Upper

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	NRBAU	Contains the upper 16 bits of the next receive buffer address to which the PCnet-PCI controller will store incoming frame data.

Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR24: Base Address of Receive Ring Lower

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	BADRL	Contains the lower 16 bits of the base address of the Receive Ring. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR25: Base Address of Receive Ring Upper

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	BADRU	Contains the upper 16 bits of the base address of the Receive Ring. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR26: Next Receive Descriptor Address Lower

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	NRDAL	Contains the lower 16 bits of the next RDRE address pointer. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR27: Next Receive Descriptor Address Upper

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	NRDAU	Contains the upper 16 bits of the next RDRE address pointer. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR28: Current Receive Descriptor Address Lower

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	CRDAL	Contains the lower 16 bits of the current RDRE address pointer. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR29: Current Receive Descriptor Address Upper

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	CRDAU	Contains the upper 16 bits of the current RDRE address pointer. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR30: Base Address of Transmit Ring Lower

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	BADXL	Contains the lower 16 bits of the base address of the Transmit Ring. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR31: Base Address of Transmit Ring Upper

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	BADXU	Contains the upper 16 bits of the base address of the Transmit Ring. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR32: Next Transmit Descriptor Address Lower

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	NXDAL	Contains the lower 16 bits of the next TDRE address pointer. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR33: Next Transmit Descriptor Address Upper

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	NXDAU	Contains the upper 16 bits of the next TDRE address pointer. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR34: Current Transmit Descriptor Address Lower

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	CXDAL	Contains the lower 16 bits of the current TDRE address pointer. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR35: Current Transmit Descriptor Address Upper

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	CXDAU	Contains the upper 16 bits of the current TDRE address pointer. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR36: Next Next Receive Descriptor Address Lower

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	NNRDAL	Contains the lower 16 bits of the next next receive descriptor address pointer. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR37: Next Next Receive Descriptor Address Upper

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	NNRDAU	Contains the upper 16 bits of the next next receive descriptor address pointer. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR38: Next Next Transmit Descriptor Address Lower

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	NNXDAL	Contains the lower 16 bits of the next next transmit descriptor address pointer. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR39: Next Next Transmit Descriptor Address Upper

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.

15–0	NNXDAU	Contains the upper 16 bits of the next next transmit descriptor address pointer. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR40: Current Receive Byte Count

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–12	RES	Reserved locations. Read and written as ZERO.
11–0	CRBC	Current Receive Byte Count. This field is a copy of the BCNT field of RMD1 of the current receive descriptor. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR41: Current Receive Status

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–8	CRST	Current Receive Status. This field is a copy of bits 31–24 of RMD1 of the current receive descriptor. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.
7–0	RES	Reserved locations. Read and written as ZERO.

CSR42: Current Transmit Byte Count

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–12	RES	Reserved locations. Read and written as ZERO.
11–0	CXBC	Current Transmit Byte Count. This field is a copy of the BCNT field of TMD1 of the current transmit descriptor. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR43: Current Transmit Status

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–8	CXST	Current Transmit Status. This field is a copy of bits 31–24 of TMD1 of the current transmit descriptor. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.
7–0	RES	Reserved locations. Read and written as ZERO.

CSR44: Next Receive Byte Count

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–12	RES	Reserved locations. Read and written as ZERO.
11–0	NRBC	Next Receive Byte Count. This field is a copy of the BCNT field of RMD1 of the next receive descriptor. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR45: Next Receive Status

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–8	NRST	Next Receive Status. This field is a copy of bits 31–24 of RMD1 of the next receive descriptor. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.
7–0	RES	Reserved locations. Read and written as ZERO.

CSR46: Poll Time Counter

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	POLL	Poll Time Counter. This counter is incremented by the PCnet-PCI controller microcode and is used to trigger the descriptor ring polling operation of the PCnet-PCI controller. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR47: Polling Interval

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	POLLINT	Polling Interval. This register contains the time that the PCnet-PCI controller will wait between successive polling operations. The POLLINT value is expressed as the twos complement of the desired interval, where each bit of POLLINT represents 1 CLK period of time. POLLINT[3:0] are ignored. (POLLINT[16] is implied to be a one, so POLLINT[15] is significant, and does not represent the sign of the twos complement POLLINT value.) The default value of this register is 0000b. This corresponds to a polling interval of 65,536 clock periods (1.966 ms when CLK = 33 MHz). The POLLINT value of 0000b is created during the microcode initialization routine, and therefore might not be seen when reading CSR47 after H_RESET or S_RESET. If the user desires to program a value for POLLINT other than the default, then the correct procedure is to first set INIT only in CSR0. Then, when the initialization sequence is complete, the user must set STOP in CSR0. Then the user may write to CSR47 and then set STRT in CSR0. In this way, the default value of 0000b in CSR47 will be overwritten with the desired user value. If the user does NOT use the standard initialization procedure (standard implies use of an initialization block in memory and setting the INIT bit of CSR0), but instead, chooses to write directly to each of the registers that are involved in the INIT operation, then it is imperative that the user also write 0000 0000 to CSR47 as part of the alternative initialization sequence. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR58: Software Style

Bit	Name	Description
		This register is an alias of the location BCR20. Accesses to/from this register are equivalent to accesses to BCR20.
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–10	RES	Reserved locations. Written as ZEROs and read as undefined.
9	CSRPCNET	CSR PCnet-ISA configuration bit. When set, this bit indicates that the PCnet-PCI controller register bits of CSR4 and CSR3 will map directly to the CSR4 and CSR3 bits of the PCnet–ISA (Am79C960) device. When cleared, this bit indicates that PCnet-PCI controller register bits of CSR4 and CSR3 will map directly to the CSR4 and CSR3 bits of the ILACC (Am79C900) device. The value of CSRPCNET is determined by the PCnet-PCI controller. CSRPCNET is read only by the host. The PCnet-PCI controller uses the setting of the Software Style register (BCR20 bits7-0/CSR58 bits 7–0) to determine the value for this bit. CSRPCNET is set by H_RESET and is not affected by S_RESET or STOP.
8	SSIZE32	Software Size 32 bits. When set, this bit indicates that the PCnet-PCI controller utilizes Am79C900 (ILACC) software structures. In particular,

Initialization Block and Transmit and Receive descriptor bit maps are affected. When cleared, this bit indicates that the PCnet-PCI controller utilizes Am79C960 (PCnet-ISA) software structures. ***Note:*** Regardless of the setting of SSIZE32, the Initialization Block must always begin on a double-word boundary.

The value of SSIZE32 is determined by the PCnet-PCI controller. SSIZE32 is read only by the host.

The PCnet-PCI controller uses the setting of the Software Style register (BCR20, bits 7-0/CSR58 bits 7-0) to determine the value for this bit. SSIZE32 is cleared by H_RESET and is not affected by S_RESET or STOP.

If SSIZE32 is reset, then bits IADR[31–24] of CSR2 will be used to generate values for the upper 8 bits of the 32 bit address bus during master accesses initiated by the PCnet-PCI controller. This action is required, since the 16-bit software structures specified by the SSIZE32=0 setting will yield only 24 bits of address for PCnet-PCI controller bus master accesses.

If SSIZE32 is set, then the software structures that are common to the PCnet-PCI controller and the host system will supply a full 32 bits for each address pointer that is needed by the PCnet-PCI controller for performing master accesses.

The value of the SSIZE32 bit has no effect on the drive of the upper 8 address bits. The upper 8 address pins are always driven, regardless of the state of the SSIZE32 bit.

Note that the setting of the SSIZE32 bit has no effect on the defined width for I/O resources. I/O resource width is determined by the state of the DWIO bit.

7–0 SWSTYLE

Software Style register. The value in this register determines the style of I/O and memory resources that are used by the PCnet-PCI controller. The Software Style selection will affect the interpretation of a few bits within the CSR space and the width of the descriptors and initialization block. Specifically: All PCnet-PCI controller CSR bits and BCR bits and all descriptor, buffer and initialization block entries not cited in the table above are unaffected by the Software Style selection and are therefore always fully functional as specified in the CSR and BCR sections.

SWSTYLE [7:0]	Style Name	CSR-PCNET	SSIZE32	Altered Bit Interpretations
00h	LANCE/ PCnet-ISA	1	0	ALL CSR4 bits will function as defined in the CSR4 section. TMD1[29] functions as ADD_FCS
01h	ILACC	0	1	CSR4[9:8], CSR4[5:4] and CSR4[1:0] will have no function, but will be writeable and readable. CSR4[15:10], CSR4[7:6] and CSR4[3:2] will function as defined in the CSR4 section. TMD1[29] becomes NO_FCS.
02h	PCnet-PCI	1	1	ALL CSR4 bits will function as defined in the CSR4 section. TMD1[29] functions as ADD_FCS
All other combs.	Res.	Undef.	Undef.	Undef.

Read/write accessible only when STOP bit is set.

The SWSTLYE register will contain the value 00h following H_RESET or S_RESET and will be unaffected by STOP.

CSR59: IR Register

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	IRREG	Reserved locations. After H_RESET, the value in this register will be 0105h. The settings of this register will have no effect on any PCnet-PCI controller function. This register always contains the same value. It is not writeable. Read accessible only when STOP bit is set.

CSR60: Previous Transmit Descriptor Address Lower

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	PXDAL	Contains the lower 16 bits of the previous TDRE address pointer. PCnet-PCI controller has the capability to stack multiple transmit frames. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR61: Previous Transmit Descriptor Address Upper

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	PXDAU	Contains the upper 16 bits of the previous TDRE address pointer. PCnet-PCI controller has the capability to stack multiple transmit frames. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR62: Previous Transmit Byte Count

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–12	RES	Reserved locations. Read and written as ZERO. Accessible only when STOP bit is set.
11–0	PXBC	Previous Transmit Byte Count. This field is a copy of the BCNT field of TMD1 of the previous transmit descriptor. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR63: Previous Transmit Status Count

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–8	PXST	Previous Transmit Status. This field is a copy of bits 31–24 of TMD1 of the previous transmit descriptor. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.
7–0	RES	Reserved locations. Read and written as ZERO. Accessible only when STOP bit is set.

CSR64: Next Transmit Buffer Address Lower

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	NXBAL	Contains the lower 16 bits of the next transmit buffer address from which the PCnet-PCI controller will transmit an outgoing frame. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR65: Next Transmit Buffer Address Upper

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	NXBAU	Contains the upper 16 bits of the next transmit buffer address from which the PCnet-PCI controller will transmit an outgoing frame. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR66: Next Transmit Byte Count

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–12	RES	Reserved locations. Read and written as ZERO. Accessible only when STOP bit is set.
11–0	NXBC	Next Transmit Byte Count. This field is a copy of the BCNT field of TMD1 of the next transmit descriptor.

		Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR67: Next Transmit Status

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–8	NXST	Next Transmit Status. This field is a copy of bits 31–24 of TMD1 of the next transmit descriptor. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.
7–0	RES	Reserved locations. Read and written as ZERO. Accessible only when STOP bit is set.

CSR72: Receive Ring Counter

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	RCVRC	Receive Ring Counter location. Contains a Two's complement binary number used to number the current receive descriptor. This counter interprets the value in CSR76 as pointing to the first descriptor. A counter value of ZERO corresponds to the last descriptor in the ring. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR74: Transmit Ring Counter

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	XMTRC	Transmit Ring Counter location. Contains a Two's complement binary number used to number the current transmit descriptor. This counter interprets the value in CSR78 as pointing to the first descriptor. A counter value of ZERO corresponds to the last descriptor in the ring. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR76: Receive Ring Length

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	RCVRL	Receive Ring Length. Contains the two's complement of the receive descriptor ring length. This register is initialized during the PCnet-PCI initialization routine based on the value in the RLEN field of the initialization block. However, this register can be manually altered. The actual receive ring length is defined by the current value in this register. The ring length can be defined as any value from 1 to 65535. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR78: Transmit Ring Length

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	XMTRL	Transmit Ring Length. Contains the two's complement of the transmit descriptor ring length. This register is initialized during the PCnet-PCI initialization routine based on the value in the TLEN field of the initialization block. However, this register can be manually altered. The actual transmit ring length is defined by the current value in this register. The ring length can be defined as any value from 1 to 65535. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR80: DMA Transfer Counter and FIFO Threshold Control

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–14	RES	Reserved locations. Read as ones and written as ZERO.

13–12 RCVFW[1:0] Receive FIFO Watermark. RCVFW controls the point at which receive DMA is requested in relation to the number of received bytes in the receive FIFO. RCVFW specifies the number of bytes which must be present (once the frame has been verified as a non-runt) before receive DMA is requested. Note however that in order for receive DMA to be performed for a new frame, at least 64 bytes must have been received. This effectively avoids having to react to receive frames which are runts or suffer a collision during the slot time (512 bit times). If the Runt Packet Accept feature is enabled, receive DMA will be requested as soon as either the RCVFW threshold is reached, or a complete valid receive frame is detected (regardless of length). RCVFW is set to a value of 10b (64 bytes) after H_RESET or S_RESET and is unaffected by STOP.

RCVFW[1:0]	Bytes Received
00	16
01	32
10	64
11	Reserved

Read/write accessible only when STOP bit is set.

Certain combinations of watermark programming and LINBC (BCR18, bits 2–0) programming may create situations where no linear bursting is possible, or where the FIFO may be excessively read or excessively written. Such combinations are declared as illegal.

Combinations of watermark settings and LINBC (BCR 18, bits 2–0) settings must obey the following relationship:

watermark (in bytes) ≥ LINBC (in bytes)

Combinations of watermark and LINBC settings that violate this rule may cause unexpected behavior.

11–10 XMTSP[1:0] Transmit Start Point. XMTSP controls the point at which preamble transmission attempts commence in relation to the number of bytes written to the transmit FIFO for the current transmit frame. When the entire frame is in the FIFO, transmission will start regardless of the value in XMTSP. XMTSP is given a value of 10b (64 bytes) after H_RESET or S_RESET and is unaffected by STOP. Regardless of XMTSP, the FIFO will not internally over write its data until at least 64 bytes (or the entire frame if < 64 bytes) have been transmitted onto the network. This ensures that for collisions within the slot time window, transmit data need not be re-written to the transmit FIFO, and re-tries will be handled autonomously by the MAC. This bit is read/write accessible only when the STOP bit is set.

XMTSP[1:0]	Bytes Written
00	4
01	16
10	64
11	112

9–8 XMTFW[1:0] Transmit FIFO Watermark. XMTFW specifies the point at which transmit DMA stops, based upon the number of write cycles that could be performed to the transmit FIFO without FIFO overflow. Transmit DMA is allowed at any time when the number of write cycles specified by XMTFW could be executed without causing transmit FIFO overflow. XMTFW is set to a value of 00b (8 cycles) after H_RESET or S_RESET and is unaffected by STOP. Read/write accessible only when STOP bit is set.

XMTFW[1:0]	Bytes Written
00	16
01	32
10	64
11	Reserved

Certain combinations of watermark programming and LINBC (BCR18, bits 2–0) programming may create situations where no linear bursting is possible, or where the FIFO may be excessively read or excessively written. Such combinations are declared as illegal.

Combinations of watermark settings and LINBC (BCR18, bits 2–0) settings must obey the following relationship:

watermark (in bytes) ≥ LINBC (in bytes)

Combinations of watermark and LINBC settings that violate this rule may cause unexpected behavior.

7–0 DMATC[7:0] DMA Transfer Counter. This counter contains the maximum allowable number of transfers to system memory that the Bus Interface Unit will perform during a single bus mastership period. The DMA Transfer Counter is not used to limit the number of transfers during Descriptor transfers. A value of ZERO will be interpreted as one transfer. During H_RESET or S_RESET a value of 16 is loaded in the DMA Transfer Counter. The value of DMATC is unaffected by the assertion of the STOP bit. If the DMAPLUS bit in CSR4 is set the DMA Transfer Counter is disabled.

When the DMA Transfer Counter times out in the middle of a linear burst, the linear burst will continue until a legal starting address is reached, and then the PCnet-PCI controller will relinquish the bus.

Therefore, if linear bursting is enabled, and the user wishes the PCnet-PCI controller to limit bus activity to desired_max transfers, then the DMA Transfer Counter should be programmed to a value of:

DMA Transfer Counter = (desired_max DIV (length of burst in transfers)) x length of burst in transfers

where DIV is the operation that yields the INTEGER portion of the ÷ operation.

Read/write accessible only when the STOP bit is set.

CSR82: Bus Activity Timer

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	DMABAT	Bus Activity Timer Register. If the TIMER bit in CSR4 is set, this register contains the maximum allowable time that PCnet-PCI controller will take up on the system bus during FIFO data transfers for a single DMA cycle. The Bus Activity Timer Register does not limit the number of transfers during Descriptor transfers. The DMABAT value is interpreted as an unsigned number with a resolution of 0.1 µs. For instance, a value of 51 µs would be programmed with a value of 510. If the TIMER bit in CSR4 is set, DMABAT is enabled and must be initialized by the user. The DMABAT register is undefined until written. If the user has NOT enabled the Linear Burst function and wishes the PCnet-PCI controller to limit bus activity to MAX_TIME microseconds, then the Burst Timer should be programmed to a value of: MAX_TIME– ((11 + 4w) x (CLK period)) where w = wait states If the user has enabled the Linear Burst function and wishes the PCnet-PCI controller to limit bus activity to MAX_TIME microseconds, then the Burst Timer should be programmed to a value of: MAX_TIME– (((3+lbs) x w + 10 + lbs) x (CLK period)) where w = wait states and lbs = linear burst size in number of transfers per sequence This is because the PCnet-PCI controller may use as much as one linear burst size plus three transfers in order to complete the

linear burst before releasing the bus.

As an example, if the linear burst size is 4 transfers, and the number of wait states for the system memory is 2, and the CLK period is 30ns and the MAX time allowed on the bus is 3 µs, then the Burst Timer should be programmed for:

MAX_TIME– (((3+lbs) x w + 10 + lbs) x (CLK period))

3 µs – (((3 + 4) x 2 +10 + 4) x (30 ns)) = 3 µs – (28 x 30 ns) = 3 – 0.84 µs = 2.16 µs.

Then, if the PCnet-PCIs Bus Activity Timer times out after 2.16 µs when the PCnet-PCI controller has completed all but the last three transfers of a linear burst, the PCnet-PCI controller may take as much as 0.84 µs to complete the bursts and release the bus. The bus release will occur at 2.16 + 0.84 = 3 µs.

A value of ZERO in the DMABAT register with the TIMER bit in CSR4 set to ONE will produce single linear burst sequences per bus master period when programmed for linear burst mode, and will yield sets of 3 transfers when not programmed for linear burst mode.

The Bus Timer register is set to a value of 00h after H_RESET or S_RESET and is unaffected by STOP.

Read/write accessible only when STOP bit is set.

CSR84: DMA Address Register Lower

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	DMABAL	DMA Address Register. This register contains the lower 16 bits of the address of system memory for the current DMA cycle. The Bus Interface Unit controls the Address Register by issuing increment commands to increment the memory address for sequential operations. The DMABA register is undefined until the first PCnet-PCI controller DMA operation. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR85: DMA Address Register Upper

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	DMABAU	DMA Address Register. This register contains the upper 16 bits of the address of system memory for the current DMA cycle. The Bus Interface Unit controls the Address Register by issuing increment commands to increment the memory address for sequential operations. The DMABA register is undefined until the first PCnet-PCI controller DMA operation. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR86: Buffer Byte Counter

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–12	RES	Reserved, Read and written with ones.
11–0	DMABC	DMA Byte Count Register. Contains the two's complement of the current size of the remaining transmit or receive buffer in bytes. This register is incremented by the Bus Interface Unit. The DMABC register is undefined until written. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR88: Chip ID Register Lower

Bit	Name	Description
31 – 28		Version. This 4-bit pattern is silicon-revision dependent.
27 – 12		Part number. The 16-bit code for the PCnet-PCI controller is 0010 0100 0011 0000b.

	Note that this code is not the same as the Device ID in the PCI configuration space.
11 – 1	Manufacturer ID. The 11-bit manufacturer code for AMD is 00000000001. This code is per the JEDEC Publication 106-A. Note that this code is not the same as the Vendor ID in the PCI configuration space.
0	Always a logic 1.

CSR89: Chip ID Register Upper

Bit	Name	Description
31 – 16		Reserved locations; Read as undefined.
15 – 12		Version. This 4-bit pattern is silicon-revision dependent.
11 – 0		Upper 12 bits of the PCnet-PCI controller part number. i.e. 0010 0100 0011b.

CSR92: Ring Length Conversion

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	RCON	Ring Length Conversion Register. This register performs a ring length conversion from an encoded value as found in the initialization block to a two's complement value used for internal counting. By writing bits 15–12 with an encoded ring length, a Two's complemented value is read. The RCON register is undefined until written. Read/write accessible only when STOP bit is set. These bits are unaffected by H_RESET, S_RESET or STOP.

CSR94: Transmit Time Domain Reflectometry Count

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–10	RES	Reserved locations. Read and written as ZERO.
9–0	XMTTDR	Time Domain Reflectometry reflects the state of an internal counter that counts from the start of transmission to the occurrence of loss of carrier. TDR is incremented at a rate of 10 MHz. Read accessible only when STOP bit is set. Write operations are ignored. XMTTDR is cleared by H_RESET or S_RESET.

CSR100: Bus Timeout

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	MERRTO	This register contains the value of the longest allowable bus latency (interval between assertion of $\overline{REQ}$ and assertion of $\overline{GNT}$) that a system may insert into a PCnet-PCI controller master transfer. If this value of bus latency is exceeded, then a MERR will be indicated in CSR0, bit 11, and an interrupt may be generated, depending upon the setting of the MERRM bit (CSR3, bit 11) and the IENA bit (CSR0, bit 6). The value in this register is interpreted as the unsigned number of XTAL1 clock periods divided 2. (i.e., the value in this register is given in 0.1 µsecond increments.) For example, the value 0200h (512 decimal) will cause a MERR to be indicated after 51.2 microseconds of bus latency. A value of ZERO will allow an infinitely long bus latency; i.e. a value of ZERO will never give a bus timeout error. This register is set to 0200h by H_RESET or S_RESET and is unaffected by $\overline{STOP}$. Read/write accessible only when STOP bit is set.

CSR112: Missed Frame Count

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	MFC	Missed Frame Count. Indicates the number of missed frames. MFC will roll over to a count of ZERO from the value 65535. The MPCO bit of CSR4 (bit 8) will be set each time that this occurs.

This register is always readable and is cleared by H_RESET or S_RESET or STOP.

A write to this register performs an increment when the ENTST bit in CSR4 is set.

CSR114: Receive Collision Count

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	RCC	Receive Collision Count. Indicates the total number of collisions encountered by the receiver since the last reset of the counter. RCC will roll over to a count of ZERO from the value 65535. The RCVCCO bit of CSR4 (bit 5) will be set each time that this occurs. The RCC value is read accessible at all times, regardless of the value of the STOP bit. Write operations are ignored. RCC is cleared by H_RESET or S_RESET or by setting the STOP bit. A write to this register performs an increment when the ENTST bit in CSR4 is set.

CSR122: Receive Frame Alignment Control

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–1	RES	Reserved locations, written as ZEROs and read as undefined.
0	RCVALGN	Receive Packet Align. When set, this bit forces the data field of ISO 8802-3 (IEEE/ANSI 802.3) packets to align to 0 MOD 4 address boundaries (i.e. DWORD (double-word) aligned addresses). It is important to note that this feature will only function correctly if all receive buffer boundaries are DWORD aligned and all receive buffers have 0 MOD 4 lengths. In order to accomplish the data alignment, the PCnet-PCI controller simply inserts two bytes of random data at the beginning of the receive packet (i.e. before the ISO 8802-3 (IEEE/ANSI 802.3) destination address field). The MCNT field reported to the receive descriptor will not include the extra two bytes. RCVALGN is cleared by H_RESET or S_RESET and is not affected by STOP. Read/write accessible only when STOP bit is set.

CSR124: Test Register 1

Bit	Name	Description
		This register is used to place the PCnet-PCI into various test modes. Only the Runt Packet Accept is an user accessible test mode. All other test modes are for AMD internal use only. ENTST must set before programming CSR124. ENTST must be reset after writing to CSR124 before writing to any other register. All bits in this register are cleared by H_RESET or S_RESET and are not affected by STOP.
31–4	RES	Reserved locations. Written as ZEROs and read as undefined.
3	RPA	Runt Packet Accept. This bit forces the PCnet-PCI controller receive logic to accept runt packets (packets shorter than 64 bytes). The state of the RPA bit can be changed only when the device is in the test mode (when ENTST bit in CSR4 is set to ONE). To enable RPA, the user must first write a ONE to the ENTST bit. Next, the user must first write a ONE to the RPA bit (CSR124, bit 3). Finally, the user must write a ZERO to the ENTST bit to take the device out of test mode operation. Once, the RPA bit has been set to ONE, the device will remain in the Runt Packet Accept mode until the RPA bit is cleared to ZERO.
2–0	RES	Reserved locations. Written as ZEROs and read as undefined.

Bus Configuration Registers

The Bus Configuration Registers (BCR) are used to program the configuration of the bus interface and other special features of the PCnet-PCI controller that are not related to the IEEE 8802-3 MAC functions. The BCRs are accessed by first setting the appropriate RAP value, and then by performing a slave access to the BDP.

All BCR registers are 16 bits in width in WIO mode and 32 bits in width in DWIO mode. The upper 16 bits of all BCR registers is undefined when in DWIO mode. These bits should be written as ZEROs and should be treated as undefined when read. The Default value given for any BCR is the value in the register after H_RESET, and is hexadecimal unless otherwise stated. BCR register values are unaffected by S_RESET and are unaffected by the assertion of the STOP bit.

Note that several registers have no default value. BCR3 and BCR8-15 are reserved and have undefined values. BCR2, BCR16, BCR17 and BCR21 are not observable without first being programmed through the EEPROM read operation or a user register write operation. Therefore the only observable values for these registers are those that have been programmed and a default value is not applicable.

BCR0, BCR1, BCR6, BCR16, BCR17, and BCR21 are reserved in the PCnet-PCI controller. These registers are used by other devices in the PCnet family. Writing to these registers have no effect on the operation of the PCnet-PCI controller.

Writes to those registers marked as “Reserved” will have no effect. Reads from these locations will produce undefined values.

BCR	MNEMONIC	Default	Description	User	EEPROM
0	MSRDA	0005h	Reserved	No	No
1	MSWRA	0005h	Reserved	No	No
2	MC	N/A*	Miscellaneous Configuration	Yes	Yes
3	Reserved	N/A	Reserved	No	No
4	LNKST	00C0h	Link Status Status (Default)	Yes	No
5	LED1	0084h	Receive Status (Default)	Yes	No
6	LED2	0088h	Reserved	Yes	No
7	LED3	0090h	Transmit Status (Default)	Yes	No
8–15	Reserved	N/A	Reserved	No	No
16	IOBASEL	N/A*	Reserved	Yes	Yes
17	IOBASEU	N/A*	Reserved	Yes	Yes
18	BSBC	2101h	Burst Size and Bus Control	Yes	Yes
19	EECAS	0002h	EEPROM Control and Status	Yes	No
20	SWS	0000h	Software Style	Yes	No
21	INTCON	N/A*	Reserved	Yes	Yes

** Registers marked with an “*” have no default value, since they are not observable without first being programmed through the EEPROM read operation or a user register write operation. Therefore, the only observable values for these registers are those that have been programmed and a default value is not applicable.*

BCR0: Master Mode Read Active

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	MSRDA	Reserved locations. After H_RESET, the value in this register will be 0005h. The settings of this register will have no effect on any PCnet-PCI controller function. Writes to this register have no effect on the operation of the PCnet-PCI controller and will not alter the value that is read.

BCR1: Master Mode Write Active

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	MSWRA	Reserved locations. After H_RESET, the value in this register will be 0005h. The settings of this register will have no effect on any PCnet-PCI controller function. Writes to this register have no effect on the operation of the PCnet-PCI controller and will not alter the value that is read.

BCR2: Miscellaneous Configuration

Bit	Name	Description
		Note that all bits in this register are programmable through the EEPROM PREAD operation.
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15	RES	Reserved location. Written and read as ZERO.
14	TMAULOOP	When set, this bit allows external loopback packets to pass onto the network through the TMAU interface, if the TMAU interface has been selected. If the TMAU interface has not been selected, then this bit has no effect. This bit is reset to ZERO by H_RESET and is unaffected by S_RESET or STOP.
13–9	RES	Reserved locations. Written and read as ZERO.
8	IESRWE	IEEE Shadow Ram Write Enable. The PCnet-PCI controller contains a shadow RAM on board for storage of the IEEE address following the serial EEPROM read operation. Accesses to APROM I/O Resources will be directed toward this RAM. When IESRWE is set to a ONE, then write access to the shadow RAM will be enabled. This bit is reset to ZERO by H_RESET and is unaffected by S_RESET or STOP.
7	RES	Reserved location. The default of this bit is zero. Writing a ONE to this bit has no effect on the operation of the PCnet-PCI controller. This reserved location is cleared by H_RESET and is unaffected by S_RESET or STOP.
6–3	RES	Reserved locations. Written and read as ZERO.
2	AWAKE	This bit selects one of two different sleep modes. If AWAKE=1 and the $\overline{\text{SLEEP}}$ pin is asserted, the PCnet-PCI controller goes into snooze mode. If AWAKE=0 and the $\overline{\text{SLEEP}}$ pin is asserted, the PCnet-PCI controller goes into coma mode. See Power Saving Modes section for more details. This bit only has meaning when the 10BASE-T network interface is selected. This bit is reset to ZERO by H_RESET and is unaffected by S_RESET or STOP.
1	ASEL	Auto Select. When set, the PCnet-PCI controller will automatically select the operating media interface port. If ASEL has been set to a ONE, then when the 10BASE-T transceiver is in the link pass state (due to receiving valid frame data and/or Link Test pulses or the DLNKTST bit is set), the 10BASE-T port will be used. If ASEL has been set to a ONE, then when the 10BASE-T port is in the link fail state, the AUI port will be used. Switching between the ports will not occur during transmission, to avoid any type of fragment generation. When ASEL is set to ONE, Link Beat Pulses will be transmitted on the 10BASE-T port, regardless of the state of Link Status. When ASEL is reset to ZERO, Link Beat Pulses will only be transmitted on the 10BASE-T

port when the PORTSEL bits of the Mode Register (CSR15) have selected 10BASE-T as the active port.

When ASEL is set to a ZERO, then the selected network port will be determined by the settings of the PORTSEL bits of CSR15.

The ASEL bit is reset to ONE by H_RESET and is unaffected by S_RESET or STOP.

The network port configurations are as follows:

PORTSEL CSR15[1:0]	ASEL (BCR2[1])	LINK Status (of 10BASE-T)	Network Port
0 X	1	Fail	AUI
0 X	1	Pass	10BASE-T
0 0	0	X	AUI
0 1	0	X	10BASE-T
1 0	X	X	Reserved
1 1	X	X	Reserved

0 XMAUSEL — Reserved location. The default value of this bit is a ZERO. Writing a ONE to this bit has no effect on the operation of the device. Existing drivers may write a ONE to this bit, but new drivers should write a ZERO to this bit.

BCR4: Link Status LED (LNKST)

Bit	Name	Description
		BCR4 controls the function(s) that the $\overline{\text{LNKST}}$ pin displays. Multiple functions can be simultaneously enabled on this LED pin. The LED display will indicate the logical OR of the enabled functions. BCR4 defaults to Link Status (LNKST) with pulse stretcher enabled (PSE = 1) and is fully programmable. The default setting after H_RESET for the LNKST register is 00C0h. The LNKST register value is unaffected by S_RESET or STOP.
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15	LEDOUT	This bit indicates the current (non-stretched) value of the LED output pin. A value of ONE in this bit indicates that the OR of the enabled signals is true. The logical value of the LEDOUT status signal is determined by the settings of the individual Status Enable bits of the LED register (Bits 6–0). This bit is READ only by the host, and is unaffected by H_RESET, S_RESET or STOP.
14	LEDPOL	LED Polarity. When this bit has the value ZERO, then the LED pin will be driven to a LOW level whenever the OR of the enabled signals is true and the LED pin will be disabled and allowed to float high whenever the OR of the enabled signals is false. (i.e. the LED output will be an Open Drain output and the output value will be the inverse of the LEDOUT status bit.) When this bit has the value ONE, then the LED pin will be driven to a HIGH level whenever the OR of the enabled signals is true and the LED pin will be driven to a LOW level whenever the OR of the enabled signals is false. (i.e. the LED output will be a Totem Pole output and the output value will be the same polarity as the LEDOUT status bit.) The setting of this bit will not affect the polarity of the LEDOUT bit for this register.
13	LEDDIS	LED Disable. This bit is used to disable the LED output. When LEDDIS has the value ONE, then the LED output will always be floated. When LEDDIS has the value ZERO, then the LED output value will be governed by the LEDOUT and LEDPOL values.
12–8	RES	Reserved locations. Write as ZEROS, read as undefined.
		A value of 0 disables the signal. A value of 1 enables the signal.
7	PSE	Pulse Stretcher Enable. Extends the LED illumination time for each new occurrence of the enabled function for this LED output. A value of 0 disables the function. A value of 1 enables the function.
6	LNKSTE	Link Status Enable. Indicates the current link status on the Twisted Pair interface. When this bit is set to one, a value of ONE will be passed to the LEDOUT signal to indicate that the link status state is PASS. A value of ZERO will be passed to the LEDOUT signal to indicate that the link status state is FAIL.

		A value of 0 disables the signal. A value of 1 enables the signal.
5	RCVME	Receive Match status Enable. Indicates receive activity on the network that has passed the address match function for this node. All address matching modes are included: Physical, Logical filtering, Promiscuous and Broadcast. A value of 0 disables the signal. A value of 1 enables the signal.
4	XMTE	Transmit status Enable. Indicates PCnet-PCI controller transmit activity. A value of 0 disables the signal. A value of 1 enables the signal.
3	RXPOLE	Receive Polarity status Enable. Indicates the current Receive Polarity condition on the Twisted Pair interface. A value of ONE indicates that the polarity of the RXD± pair has been reversed. A value of ZERO indicates that the polarity of the RXD± pair has *not* been reversed. Receive polarity indication is valid only if the T-MAU is in Link Pass state. A value of 0 disables the signal. A value of 1 enables the signal.
2	RCVE	Receive status Enable. Indicates receive activity on the network. A value of 0 disables the signal. A value of 1 enables the signal.
1	JABE	Jabber status Enable. Indicates that the PCnet-PCI controller is jabbering on the network. A value of 0 disables the signal. A value of 1 enables the signal.
0	COLE	Collision status Enable. Indicates collision activity on the network. A value of 0 disables the signal. A value of 1 enables the signal.

BCR5: LED1 Status

Bit	Name	Description
		BCR5 controls the function(s) that the LED1 pin displays. Multiple functions can be simultaneously enabled on this LED pin. The LED display will indicate the logical OR of the enabled functions. BCR1 defaults to Receive Status (RCV) with pulse stretcher enabled (PSE = 1) and is fully programmable. The default setting after H_RESET for the LED1 register is 0084h. The LED1 register value is unaffected by S_RESET or STOP.
31–16	RES	Reserved locations. Written as ZERO and read as undefined.
15	LEDOUT	This bit indicates the current (non-stretched) value of the LED output pin. A value of ONE in this bit indicates that the OR of the enabled signals is true. The logical value of the LEDOUT status signal is determined by the settings of the individual Status Enable bits of the LED register (Bits 6–0). This bit is READ only by the host, and is unaffected by H_RESET, S_RESET or STOP.
14	LEDPOL	LED Polarity. When this bit has the value ZERO, then the LED pin will be driven to a LOW level whenever the OR of the enabled signals is true and the LED pin will be disabled and allowed to float high whenever the OR of the enabled signals is false. (i.e. the LED output will be an Open Drain output and the output value will be the inverse of the LEDOUT status bit.) When this bit has the value ONE, then the LED pin will be driven to a HIGH level whenever the OR of the enabled signals is true and the LED pin will be driven to a LOW level whenever the OR of the enabled signals is false. (i.e. the LED output will be a Totem Pole output and the output value will be the same polarity as the LEDOUT status bit.) The setting of this bit will not affect the polarity of the LEDOUT bit for this register.
13	LEDDIS	LED Disable. This bit is used to disable the LED output. When LEDDIS has the value ONE, then the LED output will always be tristated. When LEDDIS has the value ZERO, then the LED output value will be governed by the LEDOUT and LEDPOL values.
12–8	RES	Reserved locations. Write as ZEROS, read as undefined.
7	PSE	A value of 0 disables the signal. A value of 1 enables the signal. Pulse Stretcher Enable. Extends the LED illumination time for

		each new occurrence of the enabled function for this LED output. A value of 0 disables the signal. A value of 1 enables the signal.
6	LNKSTE	Link Status Enable. Indicates the current link status on the Twisted Pair interface. When this bit is enabled, a value of ONE will be passed to the LEDOUT signal to indicate that the link status state is PASS. A value of ZERO will be passed to the LEDOUT signal to indicate that the link status state is FAIL. A value of 0 disables the signal. A value of 1 enables the signal.
5	RCVME	Receive Match status Enable. Indicates receive activity on the network that has passed the address match function for this node. All address matching modes are included: Physical, Logical filtering and Promiscuous. A value of 0 disables the signal. A value of 1 enables the signal.
4	XMTE	Transmit status Enable. Indicates PCnet-PCI controller transmit activity. A value of 0 disables the signal. A value of 1 enables the signal.
3	RXPOLE	Receive Polarity status Enable. Indicates the current Receive Polarity condition on the Twisted Pair interface. A value of ONE indicates that the polarity of the RXD± pair has been reversed. A value of ZERO indicates that the polarity of the RXD± pair has *not* been reversed. Receive polarity indication is valid only if the T-MAU is in the Link Pass state A value of 0 disables the signal. A value of 1 enables the signal.
2	RCVE	Receive status Enable. Indicates receive activity on the network. A value of 0 disables the signal. A value of 1 enables the signal.
1	JABE	Jabber status Enable. Indicates that the PCnet-PCI controller is jabbering on the network. A value of 0 disables the signal. A value of 1 enables the signal.
0	COLE	Collision status Enable. Indicates collision activity on the network. A value of 0 disables the signal. A value of 1 enables the signal.

BCR6: LED2 Status

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	LED2	Reserved locations. After H_RESET, the value in this register will be 0x0088h. The settings of this register will have no effect on any PCnet-PCI controller function. Writes to this register have no effect on the operation of the PCnet-PCI controller.

BCR7: LED3 Status

Bit	Name	Description
		BCR7 controls the function(s) that the LED3 pin displays. Multiple functions can be simultaneously enabled on this LED pin. The LED display will indicate the logical OR of the enabled functions. BCR7 defaults to Transmit Status (XMT) with pulse stretcher enabled (PSE = 1) and is fully programmable. The default setting after H_RESET for the LED3 register is 0090h. The LED3 register value is unaffected by S_RESET or STOP.
31–16	RES	Reserved location. Written as ZEROs and read as undefined.
15	LEDOUT	This bit indicates the current (non-stretched) value of the LED output pin. A value of ONE in this bit indicates that the OR of the enabled signals is true. The logical value of the LEDOUT status signal is determined by the settings of the individual Status Enable bits of the LED register (Bits 6–0). This bit is READ only by the host, and is unaffected by H_RESET, S_RESET or STOP.
14	LEDPOL	LED Polarity. When this bit has the value ZERO, then the LED

		pin will be driven to a LOW level whenever the OR of the enabled signals is true and the LED pin will be disabled and allowed to float high whenever the OR of the enabled signals is false. (i.e. the LED output will be an Open Drain output and the output value will be the inverse of the LEDOUT status bit.) When this bit has the value ONE, then the LED pin will be driven to a HIGH level whenever the OR of the enabled signals is true and the LED pin will be driven to a LOW level whenever the OR of the enabled signals is false. (i.e. the LED output will be a Totem Pole output and the output value will be the same polarity as the LEDOUT status bit.) The setting of this bit will not affect the polarity of the LEDOUT bit for this register.
13	LEDDIS	LED Disable. This bit is used to disable the LED output. When LEDDIS has the value ONE, then the LED output will always be tristated. When LEDDIS has the value ZERO, then the LED output value will be governed by the LEDOUT and LEDPOL values.
12–8	RES	Reserved locations. Write as ZEROS, read as undefined. A value of 0 disables the signal. A value of 1 enables the signal.
7	PSE	Pulse Stretcher Enable. Extends the LED illumination time for each new occurrence of the enabled function for this LED output. A value of 0 disables the signal. A value of 1 enables the signal.
6	LNKSTE	Link Status Enable. Indicates the current link status on the Twisted Pair interface. When this bit is enabled, a value of ONE will be passed to the LEDOUT signal to indicate that the link status state is PASS. A value of ZERO will be passed to the LEDOUT signal to indicate that the link status state is FAIL. A value of 0 disables the signal. A value of 1 enables the signal.
5	RCVME	Receive Match status Enable. Indicates receive activity on the network that has passed the address match function for this node. All address matching modes are included: Physical, Logical filtering and Promiscuous. A value of 0 disables the signal. A value of 1 enables the signal.
4	XMTE	Transmit status Enable. Indicates PCnet-PCI controller transmit activity. A value of 0 disables the signal. A value of 1 enables the signal.
3	RXPOLE	Receive Polarity status Enable. Indicates the current Receive Polarity condition on the Twisted Pair interface. A value of ONE indicates that the polarity of the RXD± pair has been reversed. A value of ZERO indicates that the polarity of the RXD± pair has *not* been reversed. Receive polarity indication is valid only if the T-MAU is in the Link Pass state. A value of 0 disables the signal. A value of 1 enables the signal.
2	RCVE	Receive status Enable. Indicates receive activity on the network. A value of 0 disables the signal. A value of 1 enables the signal.
1	JABE	Jabber status Enable. Indicates that the PCnet-PCI controller is jabbering on the network. A value of 0 disables the signal. A value of 1 enables the signal.
0		COLE Collision status Enable. Indicates collision activity on the network. A value of 0 disables the signal. A value of 1 enables the signal.

BCR16: I/O Base Address Lower

Bit	Name	Description
		Note that all bits in this register are programmable through the EEPROM PREAD operation.
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–5	IOBASEL	Reserved locations. After H_RESET, the value of these bits will be undefined. The settings of these bits will have no affect on any PCnet-PCI controller function. IOBASEL is not affected by S_RESET or STOP.

4–0	RES	Reserved locations. Written as ZEROS, read as undefined.

BCR17: I/O Base Address Upper

Bit	Name	Description
		Note that all bits in this register are programmable through the EEPROM PREAD operation.
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	IOBASEU	Reserved locations. After H_RESET, the value in this register will be undefined. The settings of this register will have no affect on any PCnet-PCI controller function. IOBASEU is not affected by S_RESET or STOP.

BCR18: Burst Size and Bus Control Register

Bit	Name	Description
		Note that all bits, except bit 7, in this register are programmable through the EEPROM PREAD operation.
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–11	RES	Reserved locations. After H_RESET, these five bits will read 00100b. The settings of these bits will have no affect on any PCnet-PCI controller function. Writes to these bits have no affect on the operation of PCnet-PCI controller. RES is set to 00100b by H_RESET and is not affected by S_RESET or STOP.
10	RES	Reserved location. Must be written as ZERO. Writing a one to this bit may cause the PCnet-PCI controller to malfunction in a system. This reserved location is cleared by H_RESET and is not affected by S_RESET or STOP.
9	RES	Reserved location. Written as ZERO and read as undefined. This reserved location is cleared by H_RESET and is not affected by S_RESET or STOP.
8	RES	Reserved bit. Must be written as a ONE. Will be read as a ONE. This reserved location is SET by H_RESET and is not affected by S_RESET or STOP.
7	DWIO	Double Word I/O. When set, this bit indicates that the PCnet-PCI controller is programmed for DWIO mode. When cleared, this bit indicates that the PCnet-PCI controller is programmed for Word I/O mode. This bit affects the I/O Resource Offset map and it affects the defined width of the PCnet-PCI controller's I/O resources. See the DWIO and WIO sections for more details. The PCnet-PCI controller will set DWIO if it detects a DWORD *write* access to offset 10h from the PCnet-PCI controller I/O base address (corresponding to the RDP resource). A double-word write access to offset 10h is the only way that the DWIO bit can be set. DWIO cannot be set by a direct write to BCR18. Once the DWIO bit has been set to a ONE, only a H_RESET can reset it to a ZERO. DWIO is read only by the host. DWIO is cleared by H_RESET and is not affected by S_RESET or STOP.
6	BREADE	Burst Read Enable. When set, this bit enables Linear Bursting during memory read accesses, where Linear Bursting is defined to mean that only the first transfer in the current bus arbitration will contain an address phase. Subsequent transfers will consist of data phases only. When cleared, this bit prevents the part from performing linear bursting during read accesses. In no case will the part linearly burst a descriptor access or an initialization access. BREADE should be set to ONE when the PCnet-PCI controller is used in a PCI bus application. The use of burst transfers guarantees maximum performance during memory read operations. BREADE is cleared by H_RESET and is not affected by S_RESET or STOP.

5 BWRITE

Burst Write Enable. When set, this bit enables Linear Bursting during memory write accesses, where Linear Bursting is defined to mean that only the first transfer in the current bus arbitration will contain an address phase. Subsequent transfers will consist of data phases only. When cleared, this bit prevents the part from performing linear bursting during write accesses. In no case will the part linearly burst a descriptor access or an initialization access.

BWRITE should be set to ONE when the PCnet-PCI controller is used in a PCI bus application. The use of burst transfers guarantees maximum performance during memory write operations.

BWRITE is cleared by H_RESET and is not affected by S_RESET or STOP.

4–3 RES

Reserved location. Written as ZEROs and read as undefined.

2–0 LINBC[2:0]

Linear Burst Count. The 3 bit value in this register sets the upper limit for the number of transfer cycles in a Linear Burst. This limit determines how often the PCnet-PCI controller will assert a new $\overline{\text{FRAME}}$ signal during linear burst transfers. Each time that the interpreted value of LINBC transfers is reached, the PCnet-PCI controller will assert a new $\overline{\text{FRAME}}$ signal with a new valid address. The LINBC value should contain only one active bit. LINBC values with more than one active bit may produce predictable results, but such values will not be compatible with future AMD network controllers.The LINBC entry is shifted by two bits before being used by the PCnet-PCI controller. For example, the value LINBC[2:0] = 010b is understood by the PCnet-PCI controller to mean 01000b = 8. Therefore, the value LINBC[2:0] = 010b will cause the PCnet-PCI controller to issue a new $\overline{\text{FRAME}}$ every 01000b = 8 transfers. The PCnet-PCI controller may linearly burst fewer than the value represented by LINBC, due to other conditions that cause the burst to end prematurely. Therefore, LINBC should be regarded as an upper limit to the length of linear burst.

Note that linear burst operation will only begin on certain addresses. The general rule for linear burst starting addresses is:

AD[31:00] MOD (LINBC x 16) = 0,

The following table illustrates all possible starting address values for all legal LINBC values. Note that AD[31:06] are don't care values for all addresses. Also note that while AD[1:0] do not physically exist within a 32 bit system (the PCnet-PCI controller always drives AD[1:0] to ZERO during the address phase to indicate a linear burst order), they are valid bits within the buffer pointer field of descriptor word 0.

LINBC[2:0]	LBS = Linear Burst Size (number of transfers)	Size of Burst (bytes)	Linear Burst Beginning Addresses AD[31:6] = (don't care) (AD[5:0] = (Hex)
1	4	16	00, 10, 20, 30
2	8	32	00, 20
4	16	64	00

There are several events which may cause early termination of linear burst. Among those events are: no more data available for transfer in either a buffer or in the FIFO or if either the DMA Transfer Counter (CSR80) or the Bus Timer Register (CSR82) times out.

Certain combinations of watermark programming and LINBC programming may create situations where no linear bursting is possible, or where the FIFO may be excessively read or excessively written. Such combinations are declared as illegal.

Combinations of watermark settings and LINBC settings must obey the following relationship:

watermark (in bytes) ≥ LINBC (in bytes)

Combinations of watermark and LINBC settings that violate this rule may cause unexpected behavior.

LINBC is set to the value of 001b by H_RESET and is not affected by S_RESET or STOP. This

gives a default linear burst length of 4 transfers = 001b x 4.

BCR19: EEPROM Control and Status Register

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15	PVALID	EEPROM Valid status bit. This bit is read only by the host. A value of ONE in this bit indicates that a PREAD operation has occurred, and that 1) there is an EEPROM connected to the PCnet-PCI controller Microwire interface pins and 2) the contents read from the EEPROM have passed the checksum verification operation. A value of ZERO in this bit indicates that the contents of the EEPROM are different from the contents of the applicable PCnet-PCI controller on-board registers and/or that the checksum for the entire 36 bytes of EEPROM is incorrect *or* that no EEPROM is connected to the Microwire interface pins. PVALID is set to ZERO during H_RESET and is unaffected by S_RESET or the STOP bit. However, following the H_RESET operation, an automatic read of the EEPROM will be performed. Just as is true for the normal PREAD command, at the end of this automatic read operation, the PVALID bit may be set to ONE. Therefore, H_RESET will set the PVALID bit to ZERO at first, but the automatic EEPROM read operation may later set PVALID to a ONE. If PVALID becomes ZERO following an EEPROM read operation (either automatically generated after H_RESET, or requested through PREAD), then all EEPROM-programmable BCR locations will be reset to their H_RESET values. The content of the APROM locations, however, will not be cleared. If no EEPROM is present at the EESK, EEDI and EEDO pins, then all attempted PREAD commands will terminate early and PVALID will NOT be set. This applies to the automatic read of the EEPROM after H_RESET as well as to host–initiated PREAD commands.
14	PREAD	EEPROM Read command bit. When this bit is set to a ONE by the host, the PVALID bit (BCR19, bit 15) will immediately be reset to a ZERO and then the PCnet-PCI controller will perform a read operation of 36 bytes from the EEPROM through the Microwire interface. The EEPROM data that is fetched during the read will be stored in the appropriate internal registers on board the PCnet-PCI controller. Upon completion of the EEPROM read operation, the PCnet-PCI controller will assert the PVALID bit. EEPROM contents will be indirectly accessible to the host through I/O read accesses to the APROM (offsets 0h through Fh) and through I/O read accesses to other EEPROM programmable registers. Note that I/O read accesses from these locations will not actually access the EEPROM itself, but instead will access the PCnet-PCI controllers internal copy of the EEPROM contents. I/O write accesses to these locations may change the PCnet-PCI controller register contents, but the EEPROM locations will not be affected. EEPROM locations may be accessed directly through BCR19. At the end of the read operation, the PREAD bit will automatically be reset to a ZERO by the PCnet-PCI controller and PVALID will bet set, provided that an EEPROM existed on the Microwire interface pins and that the checksum for the entire 36 bytes of EEPROM was correct. Note that when PREAD is set to a ONE, then the PCnet-PCI controller will no longer respond to I/O accesses directed toward it, until the PREAD operation has completed successfully. The PCnet-PCI controller will terminate these I/O accesses with the assertion of $\overline{\text{DEVSEL}}$ and $\overline{\text{STOP}}$ while $\overline{\text{TRDY}}$ is not asserted, signaling to the initiator to retry the access at a later time. If a PREAD command is given to the PCnet-PCI controller but no EEPROM is attached to the Microwire interface pins, then the

PREAD command will terminate early, the PREAD bit will be cleared to a ZERO and the PVALID bit will remain reset with a value of ZERO. This applies to the automatic read of the EEPROM after H_RESET as well as to host initiated PREAD commands. EEPROM programmable locations on board the PCnet-PCI controller will be set to their default values by such an aborted PREAD operation. For example, if the aborted PREAD operation immediately followed the H_RESET operation, then the final state of the EEPROM programmable locations will be equal to the H_RESET programming for those locations.

If a PREAD command is given to the PCnet-PCI controller and the auto-detection pin (EESK/$\overline{\text{LED1}}$) indicates that no EEPROM is present, then the EEPROM read operation will still be attempted.

Note that at the end of the H_RESET operation, a read of the EEPROM will be performed automatically. This H_RESET–generated EEPROM read function will not proceed if the auto-detection pin (EESK/$\overline{\text{LED1}}$) indicates that no EEPROM is present.

PREAD is set to ZERO during H_RESET and is unaffected by S_RESET or the STOP bit.

PREAD is only writeable when the STOP bit is set to ONE.

13 EEDET

EEPROM Detect. This bit indicates the sampled value of the EESK/$\overline{\text{LED1}}$ pin at the end of H_RESET. This value indicates whether or not an EEPROM is present at the EEPROM interface. If this bit is a ONE, it indicates that an EEPROM is present. If this bit is a ZERO, it indicates that an EEPROM is not present.

The value of this bit is determined at the end of the H_RESET operation. It is unaffected by S_RESET or the STOP bit.

This bit is not writeable. It is read only.

The following table indicates the possible combinations of EEDET and the existence of an EEPROM and the resulting operations that are possible on the EEPROM Microwire interface:

Table 10. EEDET Combinations

EEDET Value (BCR19[3])	EEPROM Connected?	Result if PREAD is Set to ONE	Result of Automatic EEPROM Read Operation Following H_RESET
0	No	EEPROM read operation is attempted. Entire read sequence will occur, checksum failure will result, PVALID is reset to ZERO.	First TWO EESK clock cycles are generated, then EEPROM read operation is aborted and PVALID is reset to ZERO.
0	Yes	EEPROM read operation is attempted. Entire read sequence will occur, checksum operation will pass, PVALID is set to ONE.	First TWO EESK clock cycles are generated, then EEPROM read operation is aborted and PVALID is reset to ZERO.
1	No	EEPROM read operation is attempted. Entire read sequence will occur, checksum failure will result, PVALID is reset to ZERO.	EEPROM read operation is attempted. Entire read sequence will occur, checksum failure will result, PVALID is reset to ZERO.
1	Yes	EEPROM read operation is attempted. Entire read sequence will occur, checksum operation will pass, PVALID is set to ONE.	EEPROM read operation is attempted. Entire read sequence will occur, checksum operation will pass, PVALID is set to ONE.

12–5	RES	Reserved locations. Written as ZERO, read as undefined.
4	EEN	EEPROM port enable. When this bit is set to a one, it causes the values of ECS, ESK and EDI to be driven onto the EECS, EESK and EEDI pins, respectively. If EEN=0 and no EEPROM read function is currently active, then EECS will be driven LOW. When EEN=0 and no EEPROM read function is currently active, EESK and EEDI pins will be driven by the LED registers BCR5 and BCR4, respectively. EEN is set to ZERO by H_RESET and is unaffected by the S_RESET or STOP bit.
3	RES	Reserved location. Written as ZERO and read as undefined.
2	ECS	EEPROM Chip Select. This bit is used to control the value of the EECS pin of the Microwire interface when the EEN bit is set to ONE and the PREAD bit is set to ZERO. If EEN = 1 and PREAD = 0 and ECS is set to a ONE, then the EECS pin will be forced to a HIGH level at the rising edge of the next CLK following bit programming. If EEN = 1 and PREAD = 0 and ECS is set to a ZERO, then the EECS pin will be forced to a LOW level at the rising edge of the next CLK following bit programming. ECS has no effect on the output value of the EECS pin unless the PREAD bit is set to ZERO and the EEN bit is set to ONE. ECS is set to ZERO by H_RESET and is not affected by S_RESET or STOP.
1	ESK	EEPROM Serial Clock. This bit and the EDI/EDO bit are used to control host access to the EEPROM. Values programmed to this bit are placed onto the EESK pin at the rising edge of the next CLK following bit programming, except when the PREAD bit is set to ONE or the EEN bit is set to ZERO. If both the ESK bit and the EDI/EDO bit values are changed during one BCR19 write operation, while EEN = 1, then setup and hold times of the EEDI pin value with respect to the EESK signal edge are not guaranteed. ESK has no effect on the EESK pin unless the PREAD bit is set to ZERO and the EEN bit is set to ONE. ESK is reset to ONE by H_RESET and is not affected by S_RESET or STOP.

Table 11. EEPROM Port Enable

$\overline{\text{RST}}$ Pin	PREAD or Auto Read in Progress	EEN	EECS	EESK	EEDI
Low	X	X	0	Z	Z
High	1	X	Active	Active	Active
High	0	1	From ECS Bit of BCR19	From ESK Bit of BCR19	From EDI Bit of BCR19
High	0	0	0	$\overline{\text{LED1}}$	$\overline{\text{LNKST}}$

0	EDI/EDO	EEPROM Data In / EEPROM Data Out. Data that is written to this bit will appear on the EEDI output of the Microwire interface, except when the PREAD bit is set to ONE or the EEN bit is set to ZERO. Data that is read from this bit reflects the value of the EEDO input of the Microwire interface. EDI/EDO has no effect on the EEDI pin unless the PREAD bit is set to ZERO and the EEN bit is set to ONE. EDI/EDO is reset to ZERO by H_RESET and is not affected by S_RESET or STOP.

BCR20: Software Style

Bit	Name	Description
		This register is an alias of the location CSR58. Accesses to/from this register are equivalent to accesses to CSR58.
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–10	RES	Reserved locations. Written as ZEROs and read as undefined.
9	CSRPCNET	CSR PCnet-ISA configuration bit. When set, this bit indicates that the PCnet-PCI controller register bits of CSR4 and CSR3 will map directly to the CSR4 and CSR3 bits of the PCnet-ISA (Am79C960) device. When cleared, this bit indicates that PCnet-PCI controller register bits of CSR4 and CSR3 will map directly to the CSR4 and CSR3 bits of the ILACC (Am79C900) device. The value of CSRPCNET is determined by the PCnet-PCI controller. CSRPCNET is read only by the host. The PCnet-PCI controller uses the setting of the Software Style register (BCR20 bits7-0/CSR58 bits 7-0) to determine the value for this bit. CSRPCNET is set by H_RESET and is not affected by S_RESET or STOP.
8	SSIZE32	Software Size 32 bits. When set, this bit indicates that the PCnet-PCI controller utilizes Am79C900 (ILACC) software structures. In particular, Initialization Block and Transmit and Receive descriptor bit maps are affected. When cleared, this bit indicates that the PCnet-PCI controller utilizes Am79C960 (PCnet-ISA) software structures. ***Note:*** Regardless of the setting of SSIZE32, the Initialization Block must always begin on a double-word boundary. The value of SSIZE32 is determined by the PCnet-PCI controller. SSIZE32 is read only by the host. The PCnet-PCI controller uses the setting of the Software Style register (BCR20, bits 7-0/CSR58 bits 7-0) to determine the value for this bit. SSIZE32 is cleared by H_RESET and is not affected by S_RESET or STOP. If SSIZE32 is reset, then bits IADR[31–24] of CSR2 will be used to generate values for the upper 8 bits of the 32 bit address bus during master accesses initiated by the PCnet-PCI controller. This action is required, since the 16-bit software structures specified by the SSIZE32=0 setting will yield only 24 bits of address for PCnet-PCI controller bus master accesses. If SSIZE32 is set, then the software structures that are common to the PCnet-PCI controller and the host system will supply a full 32 bits for each address pointer that is needed by the PCnet-PCI controller for performing master accesses. The value of the SSIZE32 bit has no effect on the drive of the upper 8 address pins. The upper 8 address pins are always driven, regardless of the state of the SSIZE32 bit. Note that the setting of the SSIZE32 bit has no effect on the defined width for I/O resources. I/O resource width is determined by the state of the DWIO bit.

7–0 SWSTYLE

Software Style register. The value in this register determines the style of I/O resources that shall be used by the PCnet-PCI controller. The Software Style selection will affect the interpretation of a few bits within the CSR space and the width of the descriptors and initialization block. Specifically:

Table 12. SWSTYLE Values

SWSTYLE [7:0]	Style Name	CSR-PCNET	SSIZE32	Altered Bit Interpretations
00h	LANCE/ PCnet-ISA	1	0	ALL CSR4 bits will function as defined in the CSR4 section. TMD1[29] functions as ADD_FCS
01h	ILACC	0	1	CSR4[9:8], CSR4[5:4] and CSR4[1:0] will have no function, but will be writeable and readable. CSR4[15:10], CSR4[7:6] and CSR4[3:2] will function as defined in the CSR4 section. TMD1[29] becomes NO_FCS.
02h	PCnet-PCI	1	1	ALL CSR4 bits will function as defined in the CSR4 section. TMD1[29] functions as ADD_FCS
All other combs.	Res.	Undef.	Undef.	Undef.

All PCnet-PCI controller CSR bits and BCR bits and all descriptor, buffer and initialization block entries not cited in the table above are unaffected by the Software Style selection and are therefore always fully functional as specified in the CSR and BCR sections.

Read/write accessible only when STOP bit is set.

The SWSTLYE register will contain the value 00h following H_RESET or S_RESET and will be unaffected by STOP.

BCR21: Interrupt Control

Bit	Name	Description
31–16	RES	Reserved locations. Written as ZEROs and read as undefined.
15–0	INTCON	Reserved locations. Writes to this register will have no effect on the operation of the PCnet-PCI controller.

Initialization Block

When SSIZE32=0 (BCR20, bit 8), then the software structures are defined to be 16 bits wide. The base address of the Initialization block in this mode must be aligned to a WORD boundary, i.e. CSR1, bit 0 and CSR16, bit 0 must be set to ZERO. When SSIZE32 = 0, the initialization block looks like Table 13.

Note that the PCnet-PCI device performs DWORD accesses to read the initialization block. This statement is always true, regardless of the setting of the SSIZE32 bit.

When SSIZE32=1 (BCR20, bit 8), then the software structures are defined to be 32 bits wide. The base address of the Initialization block in this mode must be aligned to a DOUBLE WORD boundary, i.e., CSR1, bits 0 and 1 and CSR16, bits 0 and 1 must be set to ZERO. When SSIZE32 = 1, the initialization block looks like Table 14.

Table 13. 16-Bit Data Structure Initialization Block

Address	Bits 15–13	Bit 12	Bits 11–8	Bits 7–4	Bits 3–0
IADR+00h	MODE 15–00				
IADR+02h	PADR 15–00				
IADR+04h	PADR 31–16				
IADR+06h	PADR 47–32				
IADR+08h	LADRF 15–00				
IADR+0Ah	LADRF 31–16				
IADR+0Ch	LADRF 47–32				
IADR+0Eh	LADRF 63–48				
IADR+10h	RDRA 15–00				
IADR+12h	RLEN	0	RES	RDRA 23–16	
IADR+14h	TDRA 15–00				
IADR+16h	TLEN	0	RES	TDRA 23–16	

Table 14. 32-Bit Data Structure Initialization Block

Address	Bits 31–28	Bits 27–24	Bits 23–20	Bits 19–16	Bits 15–12	Bits 11–8	Bits 7–4	Bits 3–0
IADR+00h	TLEN	RES	RLEN	RES	MODE			
IADR+04h	PADR 31–00							
IADR+08h	RES				PADR 47–32			
IADR+0Ch	LADR 31–00							
IADR+10h	LADR 63–32							
IADR+14h	RDRA 31–00							
IADR+18h	TDRA 31–00							

RLEN and TLEN

When SSIZE32=0 (BCR20, bit 8), then the software structures are defined to be 16 bits wide, and the RLEN and TLEN fields in the initialization block are 3 bits wide, occupying bits 15, 14, and 13, and the value in these fields determines the number of Receive and Transmit Descriptor Ring Entries (DRE) which are used in the descriptor rings. Their meaning is as follows:

R/TLEN	# of DREs
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

If a value other than those listed in the above table is desired, CSR76 and CSR78 can be written after initialization is complete. See the description of the appropriate CSRs.

When SSIZE32=1 (BCR20, bit 8), then the software structures are defined to be 32 bits wide, and the RLEN and TLEN fields in the initialization block are 4 bits wide, occupying bits 23–20 (RLEN) and 31–28 (TLEN) and the value in these fields determines the number of Receive and Transmit Descriptor Ring Entries (DRE) which are used in the descriptor rings. Their meaning is as follows:

R/TLEN	# of DREs
0000	1
0001	2
0010	4
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256
1001	512
11XX	512
1X1X	512

If a value other than those listed in the above table is desired, CSR76 and CSR78 can be written after initialization is complete. See the description of the appropriate CSRs.

RDRA and TDRA

TDRA and RDRA indicate where the transmit and receive descriptor rings, respectively, begin. When SSIZE32=1 (BCR20, bit 8), each DRE must be aligned to a 16-byte boundary (TDRA [3:0]=0, RDRA [3:0]=0). When SSIZE32=0 (BCR20, bit 8), each DRE must be aligned to an 8-byte boundary (TDRA [2:0]=0, RDRA [2:0]=0).

LADRF

The Logical Address Filter (LADRF) is a 64-bit mask that is used to accept incoming Logical Addresses. If the first bit in the destination address of the incoming frame (as received from the wire) is a “1”, the address is deemed logical. If the first bit is a “0”, it is a physical address and is compared against the physical address that was loaded through the initialization block.

A logical address is passed through the CRC generator, producing a 32-bit result. The high order 6 bits of the CRC is used to select one of the 64-bit positions in the Logical Address Filter. If the selected filter bit is set, the address is accepted and the frame is placed into memory.

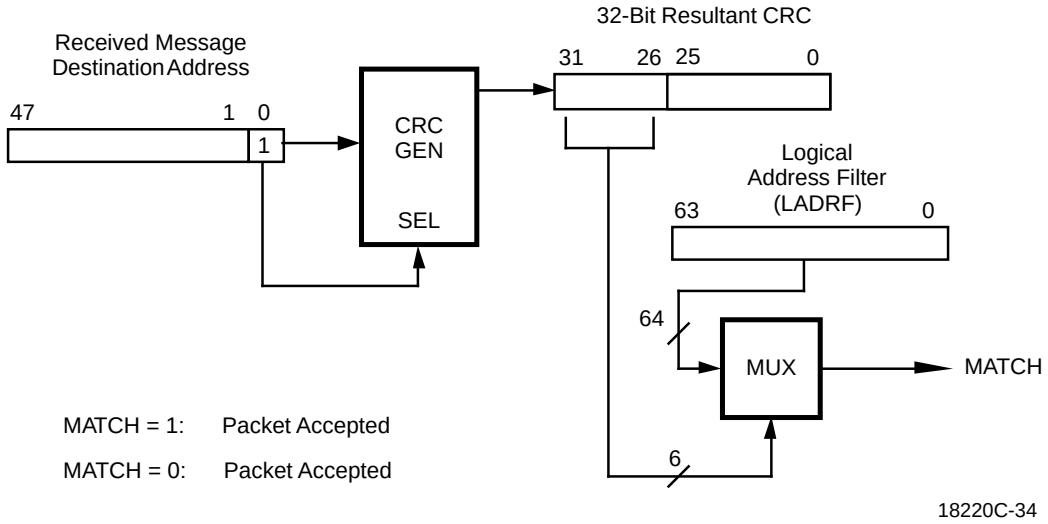


Figure 32. Address Match Logic

The Logical Address Filter is used in multicast addressing schemes. The acceptance of the incoming frame based on the filter value indicates that the message may be intended for the node. It is the node's responsibility to determine if the message is actually intended for the node by comparing the destination address of the stored message with a list of acceptable logical addresses.

If the Logical Address Filter is loaded with all ZEROs and promiscuous mode is disabled, all incoming logical addresses except broadcast will be rejected.

The Broadcast address, which is all ones, does not go through the Logical Address Filter and is handled as follows:

- If the Disable Broadcast Bit is cleared, the broadcast address is accepted.
- If the Disable Broadcast Bit is set and promiscuous mode is enabled, the broadcast address is accepted.
- If the Disable Broadcast Bit is set and promiscuous mode is disabled, the broadcast address is rejected.

If external loopback is used, the FCS logic must be allocated to the receiver (by setting the DXMTFCS bit in CSR15, and clearing the ADD_FCS bit in TMD1) when using multicast addressing.

PADR

This 48-bit value represents the unique node address assigned by the ISO 8802-3 (IEEE/ANSI 802.3) and used for internal address comparison. PADR[0] is compared with the first bit in the destination address set the incoming frame (as received from the wire) the first address bit transmitted on the wire, and must be ZERO. The six hex-digit nomenclature used by the ISO 8802-3 (IEEE/ANSI 802.3) maps to the PCnet-PCI PADR register as follows: the first byte is PADR[7:0], with PADR[0] being the least significant bit of the byte. The second ISO 8802-3 (IEEE/ANSI 802.3) byte is compared with PADR[15:8], again from LS bit to MS bit, and so on. The sixth byte is compared with PADR[47:40], the LS bit being PADR[40].

MODE

The mode register in the initialization block is copied into CSR15 and interpreted according to the description of CSR15.

Receive Descriptors

When SSIZE32=0 (BCR20, bit 8), then the software structures are defined to be 16 bits wide, and receive descriptors look like this (CRDA = Current Receive Descriptor Address):

Table 15. 16-Bit Data Structure Receive Descriptor

Address	15	14	13	12	11	10	9	8	7–0
CRDA+00h	RBADR[15:0]								
CRDA+02h	OWN	ERR	FRAM	OFLO	CRC	BUFF	STP	ENP	RBADR[23:16]
CRDA+04h	1	1	1	1	BCNT				
CRDA+06h	0	0	0	0	MCNT				

When SSIZE32=1 (BCR 20, bit 8), then the software structures are defined to be 32 bits wide, and receive descriptors look like this (CRDA = Current Receive Descriptor Address):

Table 16. 32-Bit Data Structure Receive Descriptor

Address	31	30	29	28	27	26	25	24	23–16	15–12	11–0
CRDA+00h	RBADR[31:0]										
CRDA+04h	OWN	ERR	FRAM	OFLO	CRC	BUFF	STP	ENP	RES	1111	BCNT
CRDA+08h	RCC								RPC	0000	MCNT
CRDA+0Ch	RESERVED										

RMD0

Bit	Name	Description
31–0	RBADR	Receive Buffer address. This field contains the address of the receive buffer that is associated with this descriptor.

RMD1

Bit	Name	Description
31	OWN	This bit indicates that the descriptor entry is owned by the host (OWN=0) or by the PCnet-PCI controller (OWN=1). The PCnet-PCI controller clears the OWN bit after filling the buffer pointed to by the descriptor entry. The host sets the OWN bit after emptying the buffer. Once the PCnet-PCI controller or host has relinquished ownership of a buffer, it must not change any field in the descriptor entry.
30	ERR	ERR is the OR of FRAM, OFLO, CRC, or BUFF. ERR is set by the PCnet-PCI controller and cleared by the host.
29	FRAM	FRAMING ERROR indicates that the incoming frame contained a non-integer multiple of eight bits and there was an FCS error. If there was no FCS error on the incoming frame, then FRAM will not be set even if there was a non integer multiple of eight bits in the frame. FRAM is not valid in internal loopback mode. FRAM is valid only when ENP is set and OFLO is not. FRAM is set by the PCnet-PCI controller and cleared by the host.
28	OFLO	OVERFLOW error indicates that the receiver has lost all or part of the incoming frame, due to an inability to store the frame in a memory buffer before the internal FIFO overflowed. OFLO is valid only when ENP is not set. OFLO is set by the PCnet-PCI controller and cleared by the host.
27	CRC	CRC indicates that the receiver has detected a CRC (FCS) error on the incoming frame. CRC is valid only when ENP is set and OFLO is not. CRC is set by the PCnet-PCI controller and cleared by the host.
26	BUFF	BUFFER ERROR is set any time the PCnet-PCI controller does not own the next buffer while data chaining a received frame. This can occur in either of two ways: 1. The OWN bit of the next buffer is ZERO. 2. FIFO overflow occurred before the PCnet-PCI controller received the STATUS byte (RMD1[31:24] of the next descriptor). If a Buffer Error occurs, an Overflow Error may also occur internally in the FIFO, but will not be reported in the descriptor status entry unless both BUFF and OFLO errors occur at the same time . BUFF is set by the PCnet-PCI controller and cleared by the host.
25	STP	START OF PACKET indicates that this is the first buffer used by the PCnet-PCI controller for this frame. It is used for data chaining buffers. When SPRINTEN=0 (CSR3, bit 5), STP is set by the PCnet-PCI controller and cleared by the host. When SPRINTEN=1 (CSR3, bit 5), STP must be set by the host.
24	ENP	END OF PACKET indicates that this is the last buffer used by the PCnet-PCI controller for this frame. It is used for data chaining buffers. If both STP and ENP are set, the frame fits into one buffer and there is no data chaining. ENP is set by the PCnet-PCI controller and cleared by the host.
23–16	RES	Reserved locations. These locations should be read and written as ZEROs.
15–12	ONES	These four bits must be written as ONES. They are written by the host and unchanged by the PCnet-PCI controller.
11–00	BCNT	BUFFER BYTE COUNT is the length of the buffer pointed to by this descriptor, expressed as the two's complement of the length of the buffer. This field is written by the host and unchanged by the PCnet-PCI controller.

RMD2

Bit	Name	Description
31–24	RCC	Receive Collision Count. Indicates the accumulated number of collisions on the network since the last packet was received, excluding collisions that occurred during transmissions from this node. The PCnet-PCI implementation of this counter may not be compatible with the ILACC RCC definition. If network statistics are to be monitored, then CSR114 should be used for the purpose of monitoring Receive collisions instead of these bits.
23–16	RPC	Runt Packet Count. Indicates the accumulated number of runts that were addressed to this node since the last time that a receive packet was successfully received and its corresponding RMD2 ring entry was written to by the PCnet-PCI controller. In order to be included in the RPC value, a runt must be long enough to meet the minimum requirement of the internal address matching logic. The minimum requirement for a runt to pass the internal address matching mechanism is: 18 bits of valid preamble plus a valid SFD detected, followed by 7 bytes of frame data. This requirement is unvarying, regardless of the address matching mechanisms in force at the time of reception. (i.e. physical, logical, broadcast or promiscuous). The PCnet-PCI implementation of this counter may not be compatible with the ILACC RPC definition.
15–12	ZEROS	This field is reserved. PCnet-PCI controller will write ZEROs to these locations.
11–0	MCNT	MESSAGE BYTE COUNT is the length in bytes of the received message, expressed as an unsigned binary integer. MCNT is valid only when ERR is clear and ENP is set. MCNT is written by the PCnet-PCI controller and cleared by the host.

RMD3

Bit	Name	Description
31–0	RES	Reserved locations.

Transmit Descriptors

When SSIZE32=0 (BCR 20, bit 8), then the software structures are defined to be 16 bits wide, and transmit descriptors look like this (CXDA = Current Transmit Descriptor Address):

Table 17. 16-Bit Data Structure Transmit Descriptor

Address	15	14	13	12	11	10	9	8	7–0
CXDA+00h	TBADR[15:0]								
CXDA+02h	OWN	ERR	ADD_/ NO_ FCS	MORE	ONE	DEF	STP	ENP	TBADR[23:16]
CXDA+04h	1	1	1	1	BCNT				
CXDA+06h	BUFF	UFLO	EX DEF	LCOL	LCAR	RTRY	TDR		

When SSIZE32=1 (BCR 20, bit 8), then the software structures are defined to be 32 bits wide, and transmit descriptors look like this (CXDA = Current Transmit Descriptor Address):

Table 18. 32-Bit Data Structure Transmit Descriptor

Address	31	30	29	28	27	26	25	24	23–16	15–4	3–0
CXDA+00h	TBADR[31:0]										
CXDA+04h	OWN	ERR	ADD_/ NO_ FCS	MORE	ONE	DEF	STP	ENP	RES	1111	BCNT
CXDA+08h	BUFF	UFLO	EX DEF	LCOL	LCAR	RTRY	TDR			RES	TRC
CXDA+0Ch	RESERVED										

TMD0

Bit	Name	Description
31–0	TBADR	Transmit Buffer address. This field contains the address of the transmit buffer that is associated with this descriptor.

TMD1

Bit	Name	Description
31	OWN	This bit indicates that the descriptor entry is owned by the host (OWN=0) or by the PCnet-PCI controller (OWN=1). The host sets the OWN bit after filling the buffer pointed to by the descriptor entry. The PCnet-PCI controller clears the OWN bit after transmitting the contents of the buffer. Both the PCnet-PCI controller and the host must not alter a descriptor entry after it has relinquished ownership.
30	ERR	ERR is the OR of UFLO, LCOL, LCAR, or RTRY. ERR is set by the PCnet-PCI controller and cleared by the host. This bit is set in the current descriptor when the error occurs, and therefore may be set in any descriptor of a chained buffer transmission.
29	ADD_FCS_ NO_FCS	Bit 29 functions as ADD_FCS when programmed for the default I/O style of PCnet-ISA and when programmed for the I/O style PCnet-PCI controller. Bit 29 functions as NO_FCS when programmed for the I/O style ILACC.
	ADD_FCS	ADD_FCS dynamically controls the generation of FCS on a frame by frame basis. It is valid only if the STP bit is set. When ADD_FCS is set, the state of DXMTFCS is ignored and transmitter FCS generation is activated. When ADD_FCS = 0, FCS generation is controlled by DXMTFCS. ADD_FCS is set by the host, and unchanged by the PCnet-PCI controller. This was a reserved bit in the LANCE

(Am7990). This function differs from the ILACC function for this bit.

NO_FCS — NO_FCS dynamically controls the generation of FCS on a frame by frame basis. It is valid only if the ENP bit is set. When NO_FCS is set, the state of DXMTFCS is ignored and transmitter FCS generation is deactivated. When NO_FCS = 0, FCS generation is controlled by DXMTFCS. NO_FCS is set by the host, and unchanged by the PCnet-PCI controller. This was a reserved bit in the LANCE (Am7990). This function is identical to the ILACC function for this bit

28 MORE — MORE indicates that more than one re-try was needed to transmit a frame. The value of MORE is written by the PCnet-PCI controller. This bit has meaning only if the ENP bit is set. ONE, MORE, and RTRY are mutually exclusive.

27 ONE — ONE indicates that exactly one re-try was needed to transmit a frame. ONE flag is not valid when LCOL is set. The value of the ONE bit is written by the PCnet-PCI controller. This bit has meaning only if the ENP bit is set. ONE, MORE, and RTRY are mutually exclusive.

26 DEF — DEFERRED indicates that the PCnet-PCI controller had to defer while trying to transmit a frame. This condition occurs if the channel is busy when the PCnet-PCI controller is ready to transmit. DEF is set by the PCnet-PCI controller and cleared by the host.

25 STP — START OF PACKET indicates that this is the first buffer to be used by the PCnet-PCI controller for this frame. It is used for data chaining buffers. The STP bit must be set in the first buffer of the frame, or the PCnet-PCI controller will skip over the descriptor and poll the next descriptor(s) until the OWN and STP bits are set.

STP is set by the host and unchanged by the PCnet-PCI controller.

24 ENP — END OF PACKET indicates that this is the last buffer to be used by the PCnet-PCI controller for this frame. It is used for data chaining buffers. If both STP and ENP are set, the frame fits into one buffer and there is no data chaining. ENP is set by the host and unchanged by the PCnet-PCI controller.

23–16 RES — Reserved locations.

15–12 ONES — MUST BE ONES. This field is written by the host and unchanged by the PCnet-PCI controller.

11–00 BCNT — BUFFER BYTE COUNT is the usable length of the buffer pointed to by this descriptor, expressed as the two's complement of the length of the buffer. This is the number of bytes from this buffer that will be transmitted by the PCnet-PCI controller. This field is written by the host and unchanged by the PCnet-PCI controller. There are no minimum buffer size restrictions.

TMD2

Bit	Name	Description
31	BUFF	BUFFER ERROR is set by the PCnet-PCI controller during transmission when the PCnet-32 controller does not find the ENP flag in the current buffer and does not own the next buffer. This can occur in either of two ways: 1. The OWN bit of the next buffer is ZERO. 2. FIFO underflow occurred before the PCnet-PCI controller obtained the STATUS byte (TMD1[31:24]) of the next descriptor. BUFF is set by the PCnet-PCI controller and cleared by the host. BUFF error will turn off the transmitter (CSR0, TXON = 0). If a Buffer Error occurs, an Underflow Error will also occur. BUFF is not valid when LCOL or RTRY error is set during transmit data chaining. BUFF is set by the PCnet-PCI controller and cleared by the host.
30	UFLO	UNDERFLOW ERROR indicates that the transmitter has truncated a message due to data late from memory. UFLO indicates that the FIFO has emptied

		before the end of the frame was reached. Upon UFLO error, the transmitter is turned off (CSR0, TXON = 0). UFLO is set by the PCnet-PCI controller and cleared by the host.
29	EXDEF	Excessive Deferral. Indicates that the transmitter has experienced Excessive Deferral on this transmit frame, where Excessive Deferral is defined in ISO 8802-3 (IEEE/ANSI 802.3).
28	LCOL	LATE COLLISION indicates that a collision has occurred after the slot time of the channel has elapsed. The PCnet-PCI controller does not re-try on late collisions. LCOL is set by the PCnet-PCI controller and cleared by the host.
27	LCAR	LOSS OF CARRIER is set when the carrier is lost during an PCnet-PCI controller-initiated transmission when in AUI mode. The PCnet-PCI controller does not re-try upon loss of carrier. It will continue to transmit the whole frame until done. LCAR is not valid in Internal Loopback Mode. LCAR is set by the PCnet-PCI controller and cleared by the host. In 10BASE-T mode, LCAR will be set when T-MAU is in link fail state.
26	RTRY	RETRY ERROR indicates that the transmitter has failed after 16 attempts to successfully transmit a message, due to repeated collisions on the medium. If DRTY = 1 in the MODE register, RTRY will set after 1 failed transmission attempt. RTRY is set by the PCnet-PCI controller and cleared by the host. ONE, MORE, and RTRY are mutually exclusive.
25–16	TDR	TIME DOMAIN REFLECTOMETRY reflects the state of an internal PCnet-PCI controller counter that counts at a 10 MHz rate from the start of a transmission to the occurrence of a collision or loss of carrier. This value is useful in determining the approximate distance to a cable fault. The TDR value is written by the PCnet-PCI controller and is valid only if RTRY is set. Note that 10 MHz gives very low resolution and in general has not been found to be particularly useful. This feature is here primarily to maintain full compatibility with the LANCE device (Am7990).
15–4	RES	Reserved locations.
3–0	TRC	Transmit Retry Count. Indicates the number of transmit retries of the associated packet. The maximum count is 15. However, if a RETRY error occurs, the count will roll over to ZERO. In this case only, the Transmit Retry Count value of ZERO should be interpreted as meaning 16. TRC is written by the PCnet-PCI controller into the last transmit descriptor of a frame, or when an error terminates a frame. Valid only when OWN = 0.

TMD3

Bit	Name	Description
31–0	RES	Reserved locations.

Register Summary

PCI Configuration Registers

Note: RO = read only, RW = read/write, U = undefined value

Offset	Name	Width in Bit	Access Mode	Default Value
00h	Vendor ID	16	RO	1022h
02h	Device ID	16	RO	2000h
04h	Command	16	RW	uuuu
06h	Status	16	RW	uuuu
08h	Revision ID	8	RO	00h
09h	Programming IF	8	RO	00h
0Ah	Sub-Class	8	RO	00h
0Bh	Base-Class	8	RO	02h
0Dh	Latency Timer	8	RO	00h
0Eh	Header Type	8	RO	00h
10h	Base Address	32	RW	uuuu uuuu
3Ch	Interrupt Line	8	RW	uu
3Dh	Interrupt Pin	8	RO	01h

Control and Status Registers

Note: u = undefined value, R = Running register, S = Setup register, T = Test register

RAP Addr	Symbol	Default Value After H_RESET	Comments	Use
00	CSR0	uuuu 0004	PCnet-PCI Status Register	R
01	CSR1	uuuu uuuu	IADR[15:0]: Base Address of INIT Block Lower	S
02	CSR2	uuuu uuuu	IADR[31:16]: Base Address of INIT Block Upper	S
03	CSR3	uuuu 0000	Interrupt Masks and Deferral Control	S
04	CSR4	uuuu 0115	Test and Features Control	R
05	CSR5	uuuu 0000	Reserved	T
06	CSR6	uuuu uuuu	RXTX: RX/TX Descriptor Table Lengths	T
07	CSR7	uuuu 0000	Reserved	T
08	CSR8	uuuu uuuu	LADR0: Logical Address Filter — LADRF[15:0]	T
09	CSR9	uuuu uuuu	LADR1: Logical Address Filter — LADRF[31:16]	T
10	CSR10	uuuu uuuu	LADR2: Logical Address Filter — LADRF[47:32]	T
11	CSR11	uuuu uuuu	LADR3: Logical Address Filter — LADRF[63:48]	T
12	CSR12	uuuu uuuu	PADR0: Physical Address Register — PADR[15:0]	T
13	CSR13	uuuu uuuu	PADR1: Physical Address Register — PADR[31:16]	T
14	CSR14	uuuu uuuu	PADR2: Physical Address Register — PADR[47:32]	T
15	CSR15	see reg. desc.	MODE: Mode Register	S
16	CSR16	uuuu uuuu	IADR[15:0]: Alias of CSR1	T
17	CSR17	uuuu uuuu	IADR[31:16]: Alias of CSR2	T
18	CSR18	uuuu uuuu	CRBAL: Current RCV Buffer Address Lower	T
19	CSR22	uuuu uuuu	CRBAU: Current RCV Buffer Address Upper	T
20	CSR20	uuuu uuuu	CXBAL: Current XMT Buffer Address Lower	T
21	CSR21	uuuu uuuu	CXBAU: Current XMT Buffer Address Upper	T

Control and Status Registers (continued)

RAP Addr	Symbol	Default Value After H_RESET	Comments	Use
22	CSR22	uuuu uuuu	NRBAL: Next RCV Buffer Address Lower	T
23	CSR23	uuuu uuuu	NRBAU: Next RCV Buffer Address Upper	T
24	CSR24	uuuu uuuu	BADRL: Base Address of RCV Ring Lower	S
25	CSR25	uuuu uuuu	BADRU: Base Address of RCV Ring Upper	S
26	CSR26	uuuu uuuu	NRDAL: Next RCV Descriptor Address Lower	T
27	CSR27	uuuu uuuu	NRDAU: Next RCV Descriptor Address Upper	T
28	CSR28	uuuu uuuu	CRDAL: Current RCV Descriptor Address Lower	T
29	CSR29	uuuu uuuu	CRDAU: Current RCV Descriptor Address Upper	T
30	CSR30	uuuu uuuu	BADXL: Base Address of XMT Ring Lower	S
31	CSR31	uuuu uuuu	BADXU: Base Address of XMT Ring Upper	S
32	CSR32	uuuu uuuu	NXDAL: Next XMT Descriptor Address Lower	T
33	CSR33	uuuu uuuu	NXDAU: Next XMT Descriptor Address Upper	T
34	CSR34	uuuu uuuu	CXDAL: Current XMT Descriptor Address Lower	T
35	CSR35	uuuu uuuu	CXDAU: Current XMT Descriptor Address Upper	T
36	CSR36	uuuu uuuu	NNRDAL: Next Next Receive Descriptor Address Lower	T
37	CSR37	uuuu uuuu	NNRDAU: Next Next Receive Descriptor Address Upper	T
38	CSR38	uuuu uuuu	NNXDAL: Next Next Transmit Descriptor Address Lower	T
39	CSR39	uuuu uuuu	NNXDAU: Next Next Transmit Descriptor Address Upper	T
40	CSR40	uuuu uuuu	CRBC: Current RCV Byte Count	T
41	CSR41	uuuu uuuu	CRST: Current RCV Status	T
42	CSR42	uuuu uuuu	CXBC: Current XMT Byte Count	T
43	CSR43	uuuu uuuu	CXST: Current XMT Status	T
44	CSR44	uuuu uuuu	NRBC: Next RCV Byte Count	T
45	CSR45	uuuu uuuu	NRST: Next RCV Status	T
46	CSR46	uuuu uuuu	POLL: Poll Time Counter	T
47	CSR47	uuuu uuuu	POLLINT: Polling Interval	S
48	CSR48	uuuu uuuu	Reserved	T
49	CSR49	uuuu uuuu	Reserved	T
50	CSR50	uuuu uuuu	Reserved	T
51	CSR51	uuuu uuuu	Reserved	T
52	CSR52	uuuu uuuu	Reserved	T
53	CSR53	uuuu uuuu	Reserved	T
54	CSR54	uuuu uuuu	Reserved	T
55	CSR55	uuuu uuuu	Reserved	T
56	CSR56	uuuu uuuu	Reserved	T
57	CSR57	uuuu uuuu	Reserved	T
58	CSR58	see reg. desc.	SWS: Software Style	S
59	CSR59	uuuu 0105	IR: IR Register	T
60	CSR60	uuuu uuuu	PXDAL: Previous XMT Descriptor Address Lower	T
61	CSR61	uuuu uuuu	PXDAU: Previous XMT Descriptor Address Upper	T
62	CSR62	uuuu uuuu	PXBC: Previous XMT Byte Count	T
63	CSR63	uuuu uuuu	PXST: Previous XMT Status	T
64	CSR64	uuuu uuuu	NXBA: Next XMT Buffer Address Lower	T

Control and Status Registers (continued)

RAP Addr	Symbol	Default Value After H_RESET	Comments	Use
65	CSR65	uuuu uuuu	NXBAU: Next XMT Buffer Address Upper	T
66	CSR66	uuuu uuuu	NXBC: Next XMT Byte Count	T
67	CSR67	uuuu uuuu	NXST: Next XMT Status	T
68	CSR68	uuuu uuuu	Reserved	T
69	CSR69	uuuu uuuu	Reserved	T
70	CSR70	uuuu uuuu	Reserved	T
71	CSR71	uuuu uuuu	Reserved	T
72	CSR72	uuuu uuuu	RCVRC: RCV Ring Counter	T
73	CSR73	uuuu uuuu	Reserved	T
74	CSR74	uuuu uuuu	XMTRC: XMT Ring Counter	T
75	CSR75	uuuu uuuu	Reserved	T
76	CSR76	uuuu uuuu	RCVRL: RCV Ring Length	S
77	CSR77	uuuu uuuu	Reserved	T
78	CSR78	uuuu uuuu	XMTRL: XMT Ring Length	S
79	CSR79	uuuu uuuu	Reserved	T
80	CSR80	uuuu E810	DMATCFW: DMA Transfer Counter and FIFO Threshold	S
81	CSR81	uuuu uuuu	Reserved	T
82	CSR82	uuuu 0000	DMABAT: Bus Activity Timer	S
83	CSR83	uuuu uuuu	Reserved	T
84	CSR84	uuuu uuuu	DMABAL: DMA Address Register Lower	T
85	CSR85	uuuu uuuu	DMABAU: DMA Address Register Upper	T
86	CSR86	uuuu uuuu	DMABC: Buffer Byte Counter	T
87	CSR87	uuuu uuuu	Reserved	T
88	CSR88	0242 0003	Chip ID Register Lower	T
89	CSR89	uuuu 0242	Chip ID Register Upper	T
91	CSR91	uuuu uuuu	Reserved	T
92	CSR92	uuuu uuuu	RCON: Ring Length Conversion	T
93	CSR93	uuuu uuuu	Reserved	T
94	CSR94	uuuu 0000	XMTTDR: Transmit Time Domain Reflectometry Count	T
95	CSR95	uuuu uuuu	Reserved	T
96	CSR96	uuuu uuuu	Reserved	T
97	CSR97	uuuu uuuu	Reserved	T
98	CSR98	uuuu uuuu	Reserved	T
99	CSR99	uuuu uuuu	Reserved	T
100	CSR100	uuuu 0200	MERRTO: Bus Time-Out	S
101	CSR101	uuuu uuuu	Reserved	T
102	CSR102	uuuu uuuu	Reserved	T
103	CSR103	uuuu 0105	Reserved	T
104	CSR104	uuuu uuuu	Reserved	T
105	CSR105	uuuu uuuu	Reserved	T
106	CSR106	uuuu uuuu	Reserved	T
107	CSR107	uuuu uuuu	Reserved	T

Control and Status Registers (continued)

RAP Addr	Symbol	Default Value After H_RESET	Comments	Use
108	CSR108	uuuu uuuu	Reserved	T
109	CSR109	uuuu uuuu	Reserved	T
110	CSR110	uuuu uuuu	Reserved	T
111	CSR111	uuuu uuuu	Reserved	T
112	CSR112	uuuu 0000	MFC: Missed Frame Count	R
113	CSR113	uuuu uuuu	Reserved	T
114	CSR114	uuuu 0000	RCC: Receive Collision Count	R
115	CSR115	uuuu uuuu	Reserved	T
116	CSR116	uuuu uuuu	Reserved	T
117	CSR117	uuuu uuuu	Reserved	T
118	CSR118	uuuu uuuu	Reserved	T
119	CSR119	uuuu uuuu	Reserved	T
120	CSR120	uuuu uuuu	Reserved	T
121	CSR121	uuuu uuuu	Reserved	T
122	CSR122	see reg. desc.	Receive Frame Alignment Control	S
123	CSR123	uuuu uuuu	Reserved	T
124	CSR124	see reg. desc.	Test Register 1	T
125	CSR125	uuuu uuuu	Reserved	T
126	CSR126	uuuu uuuu	Reserved	T
127	CSR127	uuuu uuuu	Reserved	T

BCR—Bus Configuration Registers

Writes to those registers marked as “Reserved” will have no effect. Reads from these locations will produce undefined values.

BCR	MNEMONIC	Default	Description	Programmability	
				User	EEPROM
0	MSRDA	0005h	Reserved	No	No
1	MSWRA	0005h	Reserved	No	No
2	MC	N/A*	Miscellaneous Configuration	Yes	Yes
3	Reserved	N/A	Reserved	No	No
4	LNKST	00C0h	Link Status (Default)	Yes	No
5	LED1	0084h	Receive Status (Default)	Yes	No
6	LED2	0088h	Reserved	Yes	No
7	LED3	0090h	Transmit Status (Default)	Yes	No
8–15	Reserved	N/A	Reserved	No	No
16	IOBASEL	N/A*	Reserved	Yes	Yes
17	IOBASEU	N/A*	Reserved	Yes	Yes
18	BSBC	2101h	Burst Size and Bus Control	Yes	Yes
19	EECAS	0002h	EEPROM Control and Status	Yes	No
20	SWS	0000h	Software Style	Yes	No
21	INTCON	N/A*	Reserved	Yes	Yes

** Registers marked with an “*” have no default value, since they are not observable without first being programmed through the EEPROM read operation. Therefore, the only observable values for these registers are those that have been programmed and a default value is not applicable.*

ABSOLUTE MAXIMUM RATINGS

Storage Temperature . . . . . –65°C to +150°C

Ambient Temperature
Under Bias . . . . . –65°C to +125°C

Supply Voltage to AV_{SS} or V_{SSB}
(AV_{DD}, V_{DD}, V_{DDB}) . . . . . –0.3 V to +6.0 V

Stresses above those listed under Absolute Maximum Ratings may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGES

Temperature (T_A) . . . . . 0°C to +70°C

Supply Voltages
(AV_{DD}, V_{DD}, V_{DDB}) . . . . . +5 V ± 5%

All Inputs within the Range:
$AV_{SS} – 0.5\ V \le V_{IN} \le AV_{DD} + 0.5\ V$,
or $V_{SS} – 0.5\ V \le V_{IN} \le V_{DD} + 0.5\ V$,
or $V_{SSB} – 0.5\ V \le V_{IN} \le V_{DDB} + 0.5\ V$

Operating ranges define those limits between which the functionality of the device is guaranteed.

DC CHARACTERISTICS over COMMERCIAL operating ranges unless otherwise specified

Parameter Symbol	Parameter Description	Test Conditions		Min	Max	Unit
Digital Input Voltage						
V_{IL}	Input LOW Voltage				0.8	V
V_{IH}	Input HIGH Voltage			2.0		V
Digital Output Voltage						
V_{OL}	Output LOW Voltage	I_{OL1} = 3 mA I_{OL2} = 6 mA			0.55	V
		I_{OL3} = 12 mA (Note 1)			0.4	V
V_{OH}	Output HIGH Voltage (Note 2)	I_{OH} = –2 mA (Note 5)		2.4		V
Digital Input Leakage Current						
I_{IL}	Input Low Leakage Current (Note 3)	V_{IN} = 0			10	µA
I_{IH}	Input High Leakage Current (Note 3)	$V_{IN} = V_{DD}, V_{DDB}$			–10	µA
Digital Output Leakage Current						
I_{OZL}	Output Low Leakage Current (Note 4)	V_{OUT} = 0.4 V			–10	µA
I_{OZH}	Output High Leakage Current (Note 4)	$V_{OUT} = V_{DD}, V_{DDB}$			10	µA
Crystal Input Current						
V_{ILX}	XTAL1 Input LOW Voltage Threshold	V_{IN} = External Clock		–0.5	0.8	V
V_{IHX}	XTAL1 Input HIGH Voltage Threshold	V_{IN} = External Clock		V_{DD} – 0.8	V_{DD} + 0.5	V
I_{ILX}	XTAL1 Input LOW Current	V_{IN} = External Clock	Active	–120	0	µA
		$V_{IN} = V_{SS}$	Sleep	–10	+10	µA
I_{IHX}	XTAL1 Input HIGH Current	V_{IN} = External Clock	Active	0	120	µA
		$V_{IN} = V_{DD}$	Sleep		400	µA
Attachment Unit Interface (AUI)						
I_{IAXD}	Input Current at DI+ and DI–	$–1\ V < V_{IN} < AV_{DD}+0.5V$		–500	+500	µA
I_{IAXC}	Input Current at CI+ and CI–	$–1\ V < V_{IN} < AV_{DD}+0.5V$		–500	+500	µA
V_{AOD}	Differential Output Voltage |(DO+)–(DO–)|	R_L = 78 Ω		630	1200	mV
V_{AODOFF}	Transmit Differential Output Idle Voltage	R_L = 78 Ω (Note 9)		–40	40	mV
I_{AODOFF}	Transmit Differential Output Idle Current	R_L = 78 Ω (Note 8)		–1	1	mA
V_{CMT}	Transmit Output Common Mode Voltage	R_L = 78 Ω		2.5	A_{VDD}	V
V_{ODI}	DO± Transmit Differential Output Voltage Imbalance	R_L = 78 Ω (Note 7)			25	mV
V_{ATH}	Receive Data Differential Input Threshold			–35	35	mV

DC CHARACTERISTICS (continued)

Parameter Symbol	Parameter Description	Test Conditions	Min	Max	Unit
V_{ASQ}	DI± and CI± Differential Input Threshold (Squelch)		–275	–160	mV
V_{IRDVD}	DI± and CI± Differential Mode Input Voltage Range		–1.5	1.5	V
V_{ICM}	DI± and CI± Input Bias Voltage	I_{IN} = 0 mA	AV_{DD}–3.0	AV_{DD}–1.0	V
V_{OPD}	DO± Undershoot Voltage at ZERO Differential on Transmit Return to ZERO (ETD)	(Note 9)		–100	mV
Twisted Pair Interface (10BASE–T)					
I_{IRXD}	Input Current at RXD±	$AV_{DD} < V_{IN} < AV_{DD}$	–500	500	µA
R_{RXD}	RXD± Differential Input Resistance		10		KΩ
V_{TIVB}	RXD±, RXD– Open Circuit Input Voltage (Bias)	I_{IN} = 0 mA	AV_{DD}–3.0	AV_{DD}–1.5	V
V_{TIDV}	Differential Mode Input Voltage Range (RXD±)	AV_{DD} = 5.0 V	–3.1	3.1	V
V_{TSQ+}	RXD Positive Squelch Threshold (peak)	Sinusoid, 5 MHz ≤ f ≤ 10 MHz	300	520	mV
V_{TSQ-}	RXD Negative Squelch Threshold (peak)	Sinusoid, 5 MHz ≤ f ≤ 10 MHz	–520	–300	mV
V_{THS+}	RXD Post–squelch Positive Threshold (peak)	Sinusoid, 5 MHz ≤ f ≤ 10 MHz	150	293	mV
V_{THS-}	RXD Post-Squelch Negative Threshold (peak)	Sinusoid, 5 MHz ≤ f ≤ 10 MHz	–293	–150	mV
V_{LTSQ+}	RXD Positive Squelch Threshold (peak)	LRT = LOW	180	312	mV
V_{LTSQ-}	RXD Negative Squelch Threshold (peak)	LRT = LOW	–312	–180	mV
V_{LTHS+}	RXD Post-Squelch Positive Threshold (peak)	LRT = LOW	90	176	mV
V_{LTHS-}	RXD Post-Squelch Negative Threshold (peak)	LRT = LOW	–176	–90	mV
V_{RXDTH}	RXD Switching Threshold	(Note 4)	–35	35	mV
V_{TXH}	TXD± and TXP± Output HIGH Voltage	V_{SS} = 0 V	V_{DD}–0.6	V_{DD}	V
V_{TXL}	TXD± and TXP± Output LOW Voltage	V_{DD} = 5 V	V_{SS}	V_{SS}+0.6	V
V_{TXI}	TXD± and TXP± Differential Output Voltage Imbalance		–40	40	mV
V_{TX}OFF	TXD± and TXP± Idle Output Voltage			40	mV
R_{TX}	TXD± , TXP± Differential Driver Output Impedance	(Note 4)	40	80	Ω

DC CHARACTERISTICS (continued)

Parameter Symbol	Parameter Description	Test Conditions	Min	Max	Unit
Power Supply Current					
I_{DD}	Active Power Supply Current	XTAL1 = 20 MHz, CLK = 33 MHz		150	mA
I_{DDCOMA}	Sleep Mode Power Supply Current	$\overline{SLEEP}$ Active		TBD	µA
$I_{DDSNOOZE}$	Auto Wake Mode Power Supply Current	Awake Bit Set Active		TBD	µA
Pin Capacitance					
C_{IN}	Input Pin Capacitance	FC = 1 MHz (Notes 6 & 10)		10	pF
C_O	I/O or Output Pin Capacitance	FC = 1 MHz (Note 6)		10	pF
C_{CLK}	CLK Pin Capacitance	FC = 1 MHz (Note 6)		12	pF

Notes:

1. *I_{OL1} applies to AD[31:00], C/$\overline{BE}$[3:0], $\overline{REQ}$, and PAR*
   I_{OL2} applies to $\overline{FRAME}$, $\overline{IRDY}$, $\overline{TRDY}$, $\overline{DEVSEL}$, $\overline{STOP}$, $\overline{SERR}$, $\overline{PERR}$, and $\overline{LOCK}$
   I_{OL3} applies to EESK/$\overline{LED1}$, EEDO/$\overline{LED3}$, and EEDI/$\overline{LNKST}$
2. *V_{OH} does not apply to open–drain output pins.*
3. *I_{IL} and I_{IH} apply to all input pins except XTAL1.*
4. *I_{OZL} and I_{OZH} apply to all three-state output pins and bi-directional pins.*
5. *Outputs are CMOS and will be driven to rail if the load is not resistive.*
6. *Parameter not tested. Value determined by characterization.*
7. *Tested, but to values in excess of limits. Test accuracy not sufficient to allow screening guard bands.*
8. *Correlated to other tested parameters – not tested directly.*
9. *Test not implemented to data sheet specification.*
10. *C_{IN} = 8 pF for IDSEL input.*

SWITCHING CHARACTERISTICS: Bus Interface

Parameter Symbol	Parameter Description	Test Conditions	Min	Max	Unit
Clock Timing					
	CLK Frequency		0	33	MHz
t_{CYC}	CLK Period	@ 1.5 V	30	∞	ns
t_{HIGH}	CLK High Time	@ 2.0 V	12		ns
t_{LOW}	CLK Low Time	@ 0.8 V	12		ns
t_{FALL}	CLK Fall Time	over 2 V p-p	1	4	V/ns
t_{RISE}	CLK Rise Time	over 2 V p-p	1	4	V/ns
Output and Float Delay Timing					
t_{VAL}	AD[31:00], C/$\overline{BE}$[3:0], PAR, $\overline{FRAME}$, $\overline{IRDY}$, $\overline{TRDY}$, $\overline{STOP}$, $\overline{LOCK}$, $\overline{DEVSEL}$, $\overline{PERR}$, $\overline{SERR}$ Valid Delay		2	11	ns
t_{VAL} ($\overline{REQ}$)	$\overline{REQ}$ Valid Delay		1	12	ns
f_{EESK}	EESK Frequency	(See note below)		650	KHz
t_{VAL} (EEDI)	EEDI Valid Output Delay from EESK	(See note below)	100	400	ns
t_{VAL} (EESK)	EECS Valid Output Delay from EESK	(See note below)	0	400	ns
t_{ON}	AD[31:00], C/$\overline{BE}$[3:0], PAR, $\overline{FRAME}$, $\overline{IRDY}$, $\overline{TRDY}$, $\overline{STOP}$, $\overline{LOCK}$, $\overline{DEVSEL}$ Active Delay		2	11	ns
t_{OFF}	AD[31:00], C/$\overline{BE}$[3:0], PAR, $\overline{FRAME}$, $\overline{IRDY}$, $\overline{TRDY}$, $\overline{STOP}$, $\overline{LOCK}$, $\overline{DEVSEL}$ Float Delay			28	ns
Setup and Hold Timing					
t_{SU}	AD[31:00], C/$\overline{BE}$[3:0], PAR, $\overline{FRAME}$, $\overline{IRDY}$, $\overline{TRDY}$, $\overline{STOP}$, $\overline{LOCK}$, $\overline{DEVSEL}$, IDSEL Setup Time		7		ns
t_H	AD[31:00], C/$\overline{BE}$[3:0], PAR, $\overline{FRAME}$, $\overline{IRDY}$, $\overline{TRDY}$, $\overline{STOP}$, $\overline{LOCK}$, $\overline{DEVSEL}$, IDSEL Hold Time		0		ns
t_{SU} ($\overline{GNT}$)	$\overline{GNT}$ Setup Time		10		ns
t_H ($\overline{GNT}$)	$\overline{GNT}$ Hold Time		0		ns
t_{SU} (EEDO)	EEDO Setup Time to EESK	(See note below)	50		ns
t_H (EEDO)	EEDO Hold Time from EESK	(See note below)	0		ns

Note:

Parameter value is given for automatic EEPROM read operation. When EEPROM port (BCR19) is used to access the EEPROM, software is responsible for meeting EEPROM timing requirements.

SWITCHING CHARACTERISTICS: 10BASE-T Interface

Parameter Symbol	Parameter Description	Test Conditions	Min	Max	Unit
Transmit Timing					
t_{TETD}	Transmit Start of Idle		250	350	ns
t_{TR}	Transmitter rise time	(10% to 90%)		5.5	ns
t_{TF}	Transmitter fall time	(90% to 10%)		5.5	ns
t_{TM}	Transmitter rise and fall time mismatch	(t_{TM} = | t_{TR} – t_{TF}|)		1	ns
t_{PERLP}	Idle Signal Period		8	24	ms
t_{PWLP}	Idle Link Pulse Width	(See note below)	75	120	ns
t_{PWPLP}	Predistortion Idle Link Pulse Width	(See note below)	45	55	ns
t_{JA}	Transmit jabber activation time		20	150	ms
t_{JR}	Transmit jabber reset time		250	750	ms
t_{JREC}	Transmit jabber recovery time (minimum time gap between transmitted frames to prevent jabber activation)		1.0		µs
Receiving Timing					
t_{PWNRD}	RXD pulse width not to turn off internal carrier sense	$V_{IN} > V_{THS}$ (min)	136		ns
t_{PWROFF}	RXD pulse width to turn off	$V_{IN} > V_{THS}$ (min)		200	ns
t_{RETD}	Receive Start of Idle		200		ns

Note:

Not tested; parameter guaranteed by characterization.

SWITCHING CHARACTERISTICS: Attachment Unit Interface

Parameter Symbol	Parameter Description	Test Conditions	Min	Max	Unit
AUI Port					
t_{DOTR}	DO+, DO– Rise Time (10% to 90%)		2.5	5.0	ns
t_{DOTF}	DO+, DO– Fall Time (10% to 90%)		2.5	5.0	ns
t_{DORM}	DO+, DO– Rise and Fall Time Mismatch			1.0	ns
t_{DOETD}	DO± End of Transmission		200	375	ns
t_{PWODI}	DI Pulse Width Accept/Reject Threshold	$\|V_{IN}\| > \|VASQ\|$ (Note 1)	15	45	ns
t_{PWKDI}	DI Pulse Width Maintain/Turn-Off Threshold	$\|V_{IN}\| > \|VASQ\|$ (Note 2)	136	200	ns
t_{PWOCI}	CI Pulse Width Accepth/Reject Threshold	$\|V_{IN}\| > \|VASQ\|$ (Note 3)	10	26	ns
t_{PWKCI}	CI Pulse Width Maintain/Turn-Off Threshold	$\|V_{IN}\| > \|VASQ\|$ (Note 4)	90	160	ns
Internal MENDEC Clock Timing					
tx1	XTAL1 Period	V_{IN} = External Clock	49.995	50.001	ns
tx1H	XTAL1 HIGH Pulse Width	V_{IN} = External Clock	20		ns
tx1L	XTAL1 LOW Pulse Width	V_{IN} = External Clock	20		ns
tx1R	XTAL1 Rise Time	V_{IN} = External Clock		5	ns
tx1F	XTAL1 Fall Time	V_{IN} = External Clock		5	ns

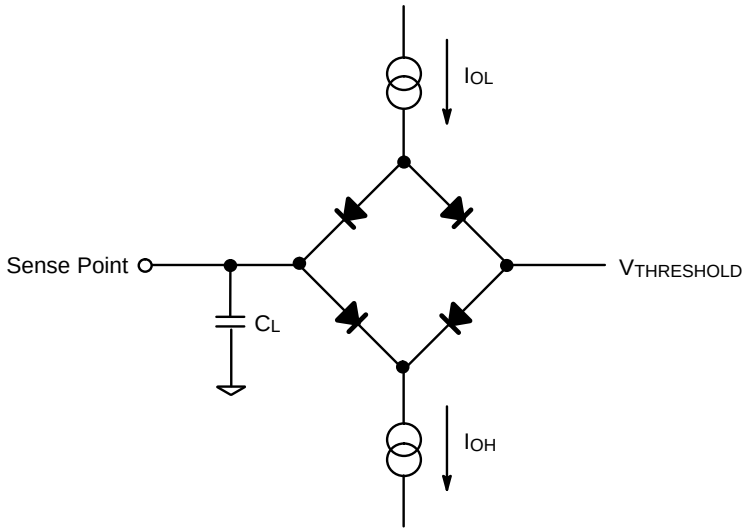
Notes:

1. *DI pulses narrower than t_{PWODI} (min) will be rejected; pulses wider than t_{PWODI} (max) will turn internal DI carrier sense on.*
2. *DI pulses narrower than t_{PWKDI} (min) will maintain internal DI carrier sense on; pulses wider than t_{PWKDI} (max) will turn internal DI carrier sense off.*
3. *CI pulses narrower than t_{PWOCI} (min) will be rejected; pulses wider than t_{PWOCI} (max) will turn internal CI carrier sense on.*
4. *CI pulses narrower than t_{PWKCI} (min) will maintain internal CI carrier sense on; pulses wider than t_{PWKCI} (max) will turn internal CI carrier sense off.*

KEY TO SWITCHING WAVEFORMS

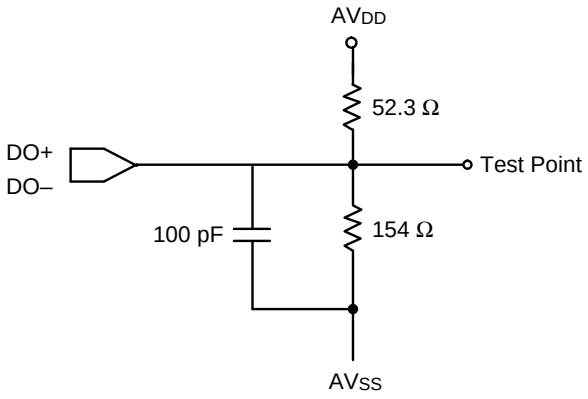
WAVEFORM	INPUTS	OUTPUTS
	Must be Steady	Will be Steady
	May Change from H to L	Will be Changing from H to L
	May Change from L to H	Will be Changing from L to H
	Don't Care, Any Change Permitted	Changing, State Unknown
	Does Not Apply	Center Line is High-Impedance "Off" State

SWITCHING TEST CIRCUITS

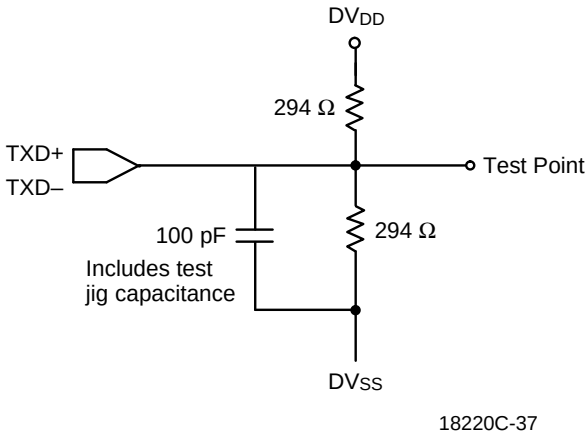


Normal and Three-State Outputs

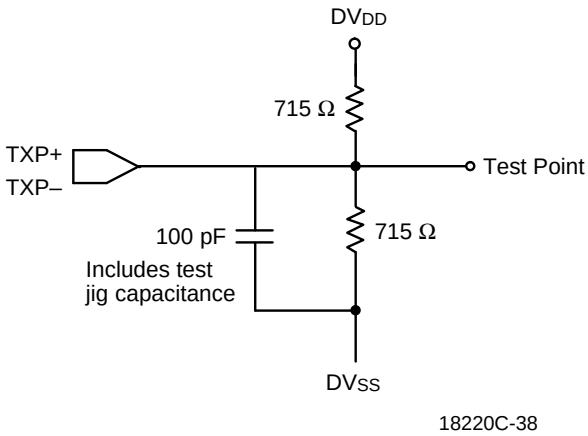
SWITCHING TEST CIRCUITS



AUI DO Switching Test Circuit

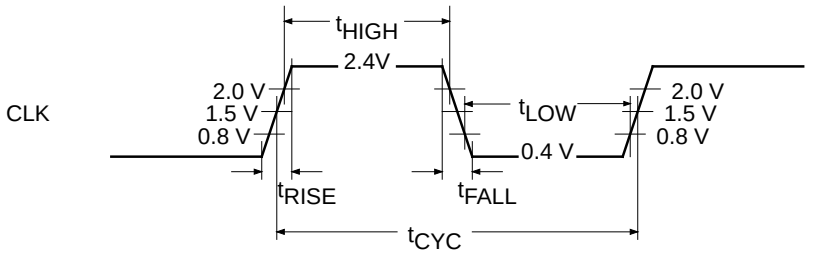


TXD Switching Test Circuit

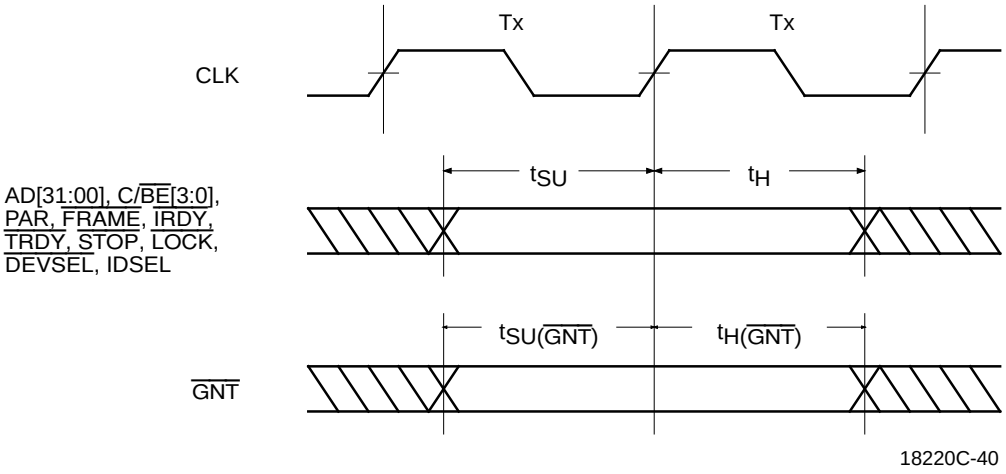


TXP Outputs Test Circuit

SWITCHING WAVEFORMS: System Bus Interface

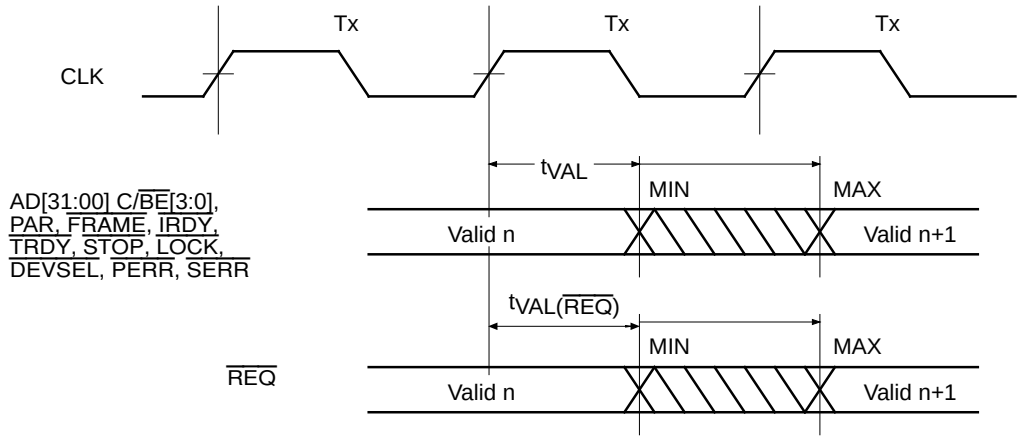


CLK Waveform



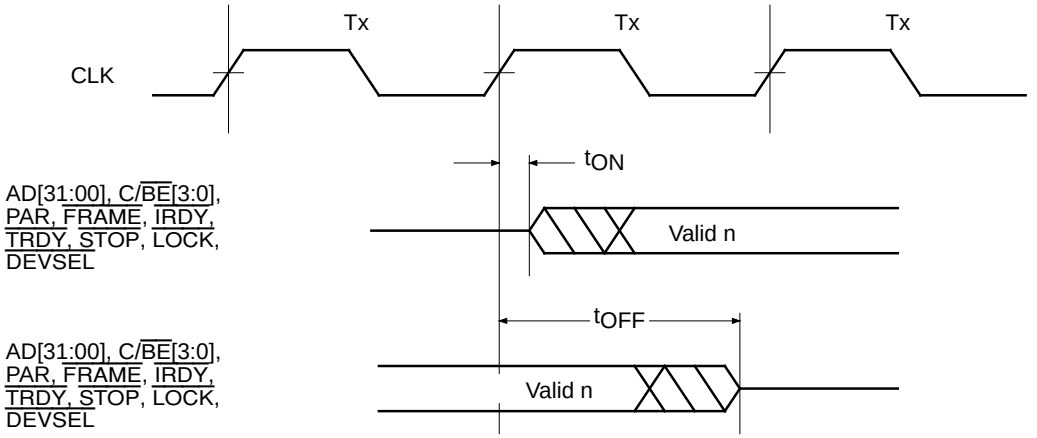
Input Setup and Hold Timing

SWITCHING WAVEFORMS: System Bus Interface



18220C-41

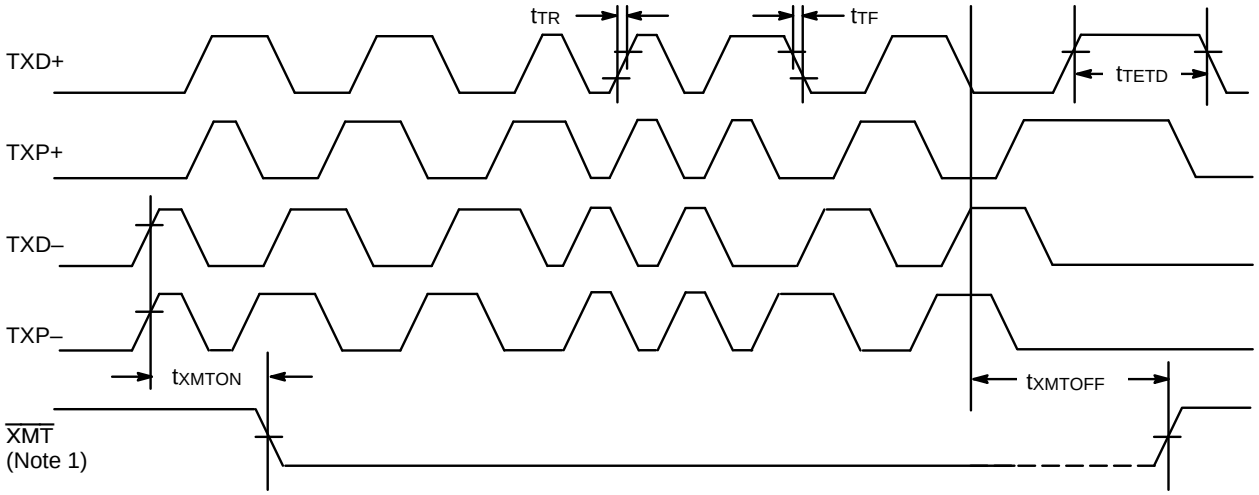
Output Valid Delay Timing



18220C-42

Output Tri–State Delay Timing

SWITCHING WAVEFORMS: 10BASE-T Interface

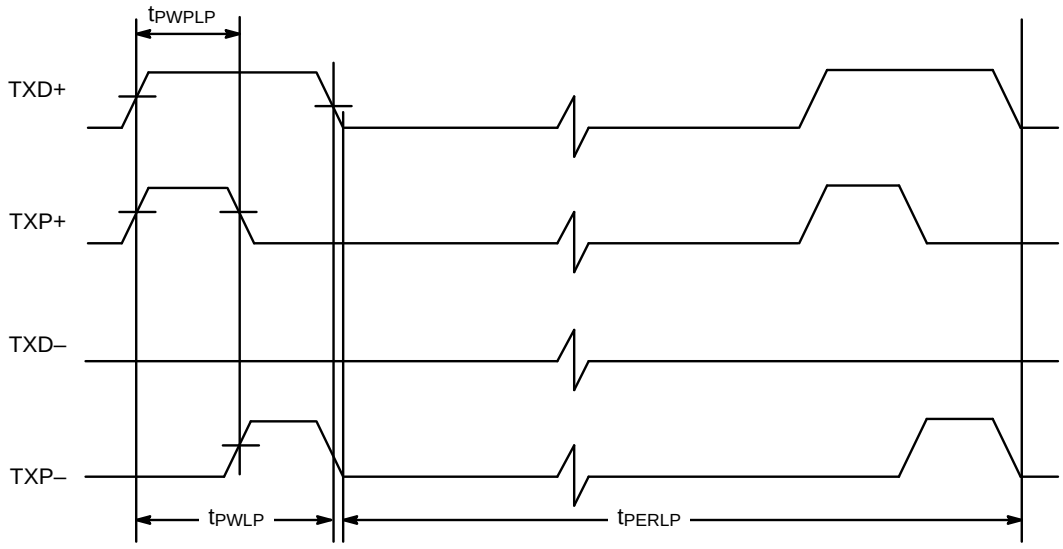


18220C-43

Note:

1. Internal signal and is shown for clarification only.

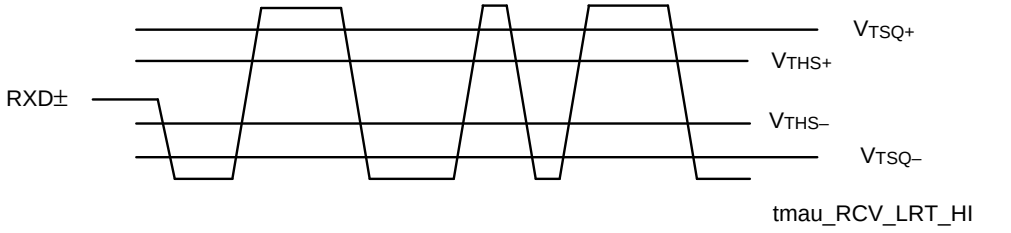
Transmit Timing



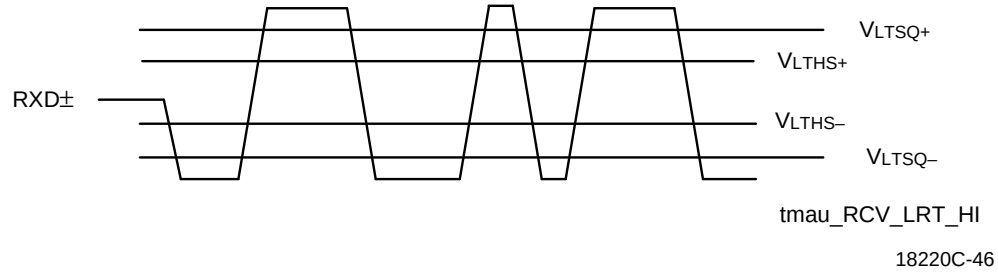
18220C-44

Idle Link Test Pulse

SWITCHING WAVEFORMS: 10BASE-T Interface

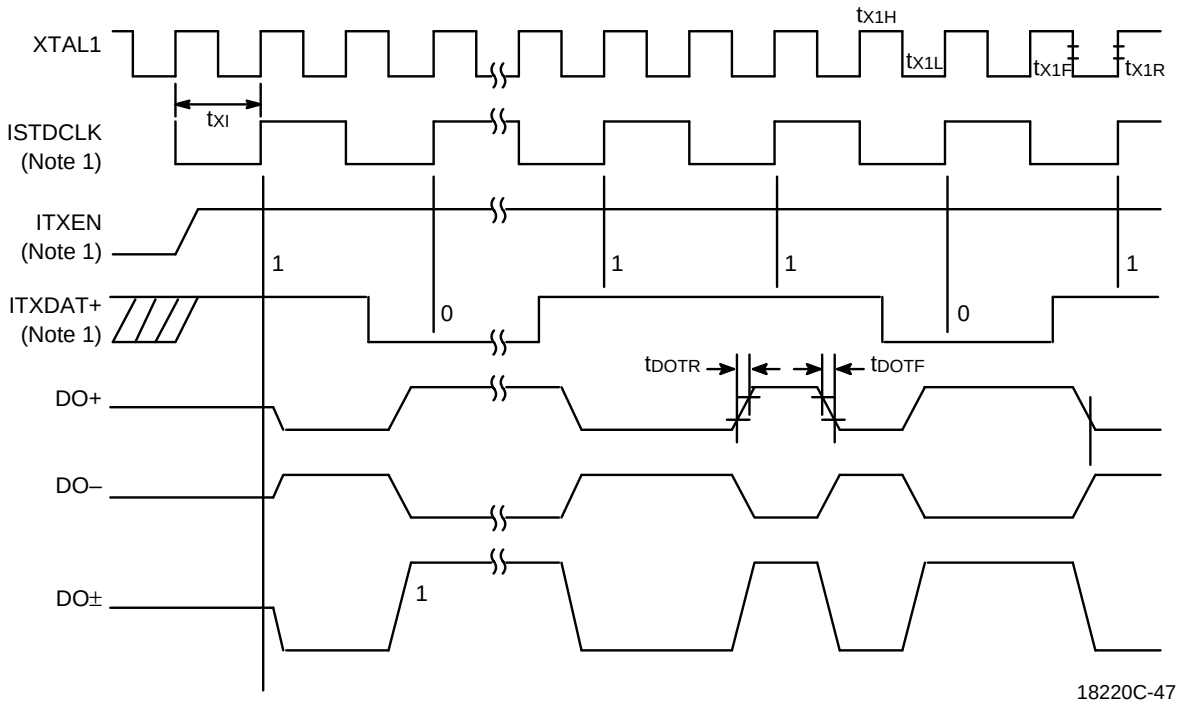


Receive Thresholds (LRT=0)



Receive Thresholds (LRT=1)

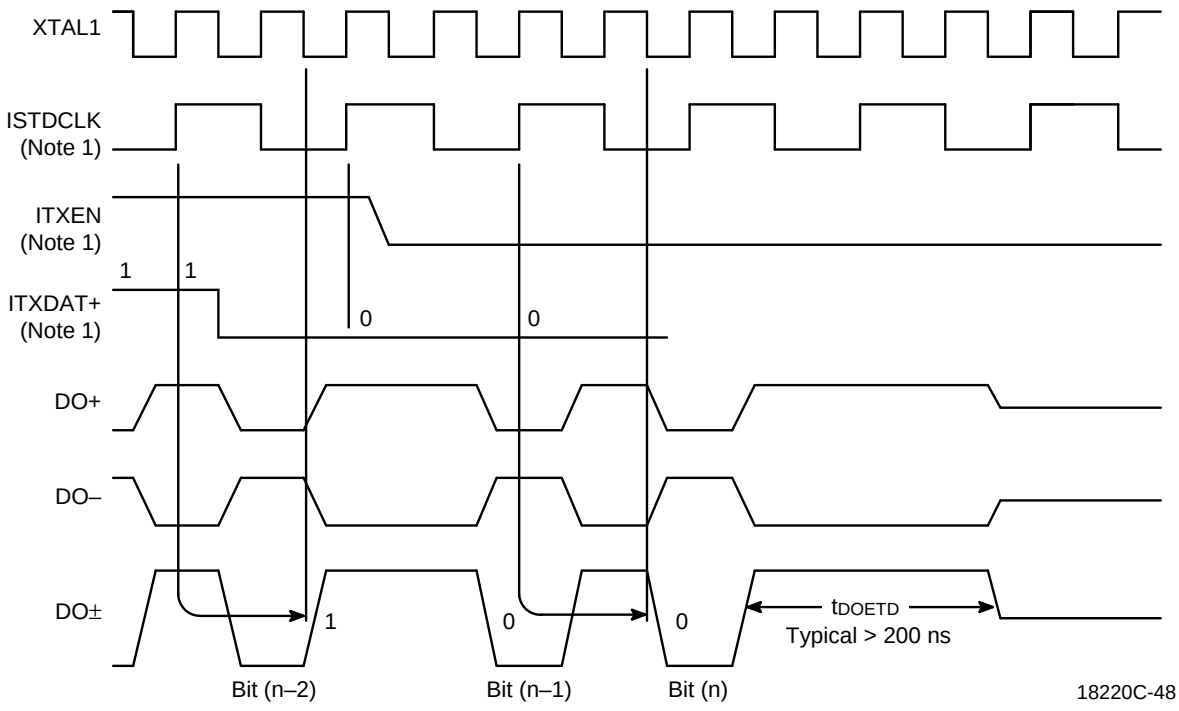
SWITCHING WAVEFORMS: Attachment Unit Interface



Note:

1. Internal signal and is shown for clarification only.

Transmit Timing – Start of Frame

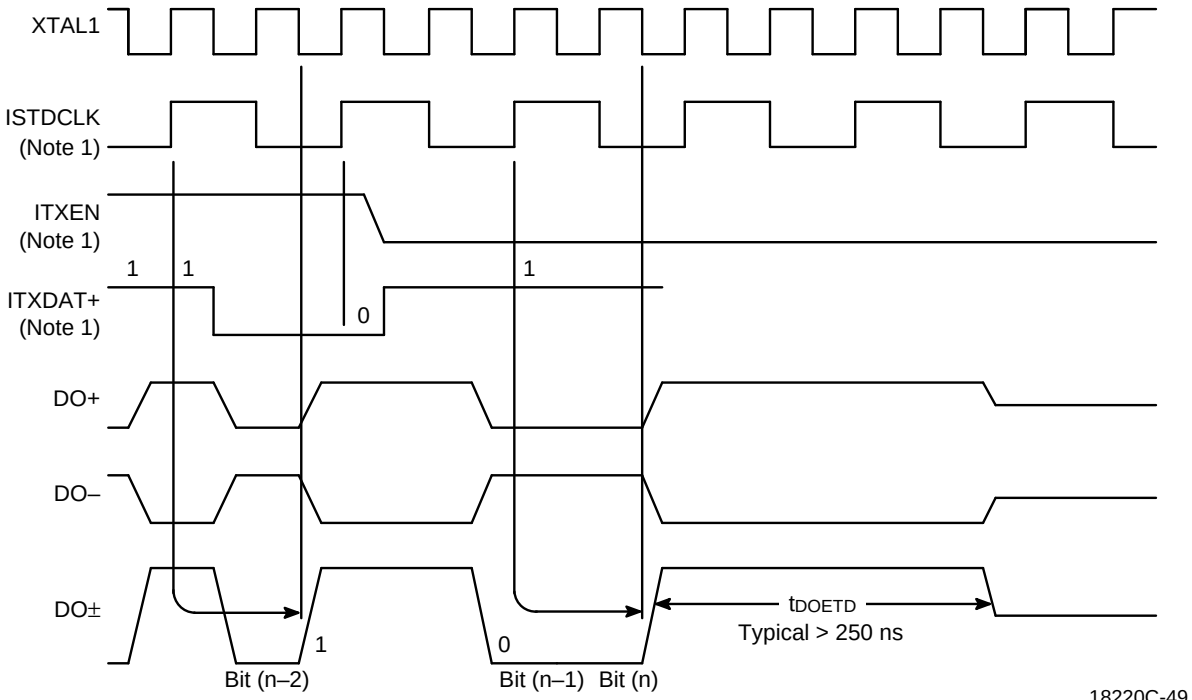


Note:

1. Internal signal and is shown for clarification only.

Transmit Timing – End of Frame (Last Bit = 0)

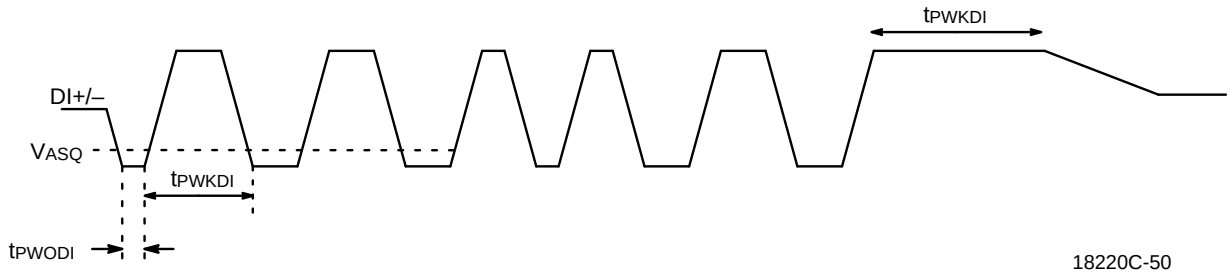
SWITCHING WAVEFORMS: Attachment Unit Interface



Note:

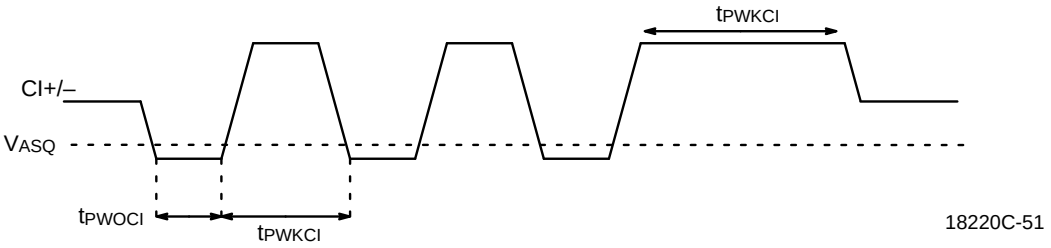
1. Internal signal and is shown for clarification only.

Transmit Timing – End of Frame (Last Bit = 1)

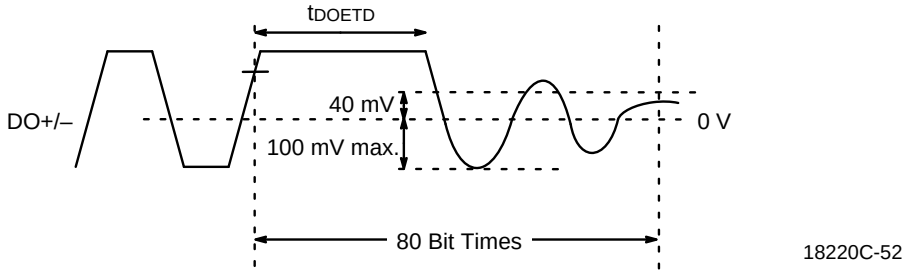


Receive Timing

SWITCHING WAVEFORMS: Attachment Unit Interface



Collision Timing



Port DO ETD Waveform

PCnet-PCI Compatible Media Interface Modules

PCnet-PCI Compatible 10BASE-T Filters and Transformers

The table below provides a sample list of PCnet-PCI compatible 10BASE-T filter and transformer modules available from various vendors. Contact the respective manufacturer for a complete and updated listing of components.

Manufacturer	Part #	Package	Filters and Transformers	Filters Transformers and Choke	Filters Transformers Dual Chokes	Filters Transformers Resistors Dual Chokes
Bel Fuse	A556-2006-DE	16-pin 0.3 DIL	√			
Bel Fuse	0556-2006-00	14-pin SIP	√			
Bel Fuse	0556-2006-01	14-pin SIP			√	
Bel Fuse	0556-6392-00	16-pin 0.5 DIL			√	
Halo Electronics	FD02-101G	16-pin 0.3 DIL	√			
Halo Electronics	FD12-101G	16-pin 0.3 DIL		√		
Halo Electronics	FD22-101G	16-pin 0.3 DIL			√	
PCA Electronics	EPA1990A	16-pin 0.3 DIL	√			
PCA Electronics	EPA2013D	16-pin 0.3 DIL		√		
PCA Electronics	EPA2162	16-pin 0.3 SIP			√	
Pulse Engineering	PE-65421	16-pin 0.3 DIL	√			
Pulse Engineering	PE-65434	16-pin 0.3 SIL			√	
Pulse Engineering	PE-65445	16-pin 0.3 DIL			√	
Pulse Engineering	PE-65467	12-pin 0.5 SMT				√
Valor Electronics	PT3877	16-pin 0.3 DIL	√			
Valor Electronics	FL1043	16-pin 0.3 DIL			√	

PCnet-PCI Compatible AUI Isolation Transformers

The table below provides a sample list of PCnet-PCI compatible AUI isolation transformers available from various vendors. Contact the respective manufacturer for a complete and updated listing of components.

Manufacturer	Part #	Package	Description
Bel Fuse	A553-0506-AB	16-pin 0.3 DIL	50 µH
Bel Fuse	S553-0756-AE	16-pin 0.3 SMD	75 µH
Halo Electronics	TD01-0756K	16-pin 0.3 DIL	75 µH
Halo Electronics	TG01-0756W	16-pin 0.3 SMD	75 µH
PCA Electronics	EP9531-4	16-pin 0.3 DIL	50 µH
Pulse Engineering	PE64106	16-pin 0.3 DIL	50 µH
Pulse Engineering	PE65723	16-pin 0.3 SMT	75 µH
Valor Electronics	LT6032	16-pin 0.3 DIL	75 µH
Valor Electronics	ST7032	16-pin 0.3 SMD	75 µH

PCnet-PCI Compatible DC/DC Converters

The table below provides a sample list of PCnet-PCI compatible DC/DC converters available from various vendors. Contact the respective manufacturer for a complete and updated listing of components.

Manufacturer	Part #	Package	Voltage	Remote On/Off
Halo Electronics	DCU0-0509D	24-pin DIP	5/-9	No
Halo Electronics	DCU0-0509E	24-pin DIP	5/-9	Yes
PCA Electronics	EPC1007P	24-pin DIP	5/-9	No
PCA Electronics	EPC1054P	24-pin DIP	5/-9	Yes
PCA Electronics	EPC1078	24-pin DIP	5/-9	Yes
Valor Electronics	PM7202	24-pin DIP	5/-9	No
Valor Electronics	PM7222	24-pin DIP	5/-9	Yes

MANUFACTURER CONTACT INFORMATION

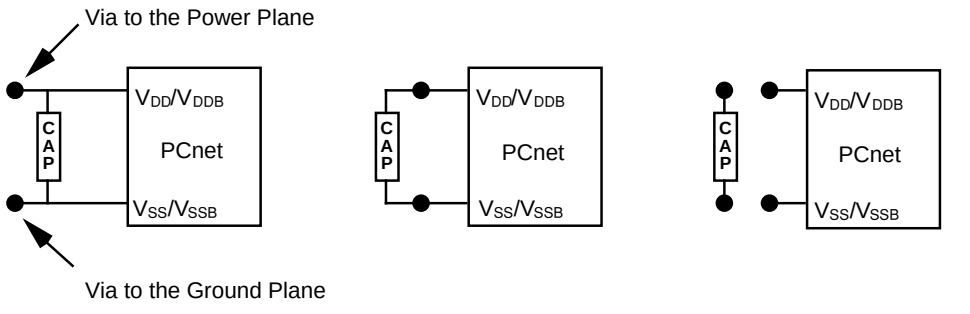
Contact the following companies for further information on their products.

Company	U.S. and Domestic		Asia	Europe
Bel Fuse	Phone:	(201) 432-0463	852-328-5515	33-1-69410402
	FAX:	(201) 432-9542	852-352-3706	33-1-69413320
Halo Electronics	Phone:	(415) 969-7313	65-285-1566	
	FAX:	(415) 367-7158	65-284-9466	
PCA Electronics	Phone:	(818) 892-0761	852-553-0165	33-1-44894800
(HPC in Hong Kong)	FAX:	(818) 894-5791	852-873-1550	33-1-42051579
Pulse Engineering	Phone:	(619) 674-8100	852-425-1651	353-093-24107
	FAX:	(619) 675-8262	852-480-5974	353-093-24459
Valor Electronics	Phone:	(619) 537-2500	852-513-8210	49-89-6923122
	FAX:	(619) 537-2525	852-513-8214	49-89-6926542

Recommendation for Power and Ground Decoupling

The mixed analog/digital circuitry in the PCnet-PCI make it imperative to provide noise-free power and ground connections to the device. Without clean power and ground connections, a design may suffer from high bit error rates or may not function at all. Hence, it is highly recommended that the guidelines presented here are followed to ensure a reliable design.

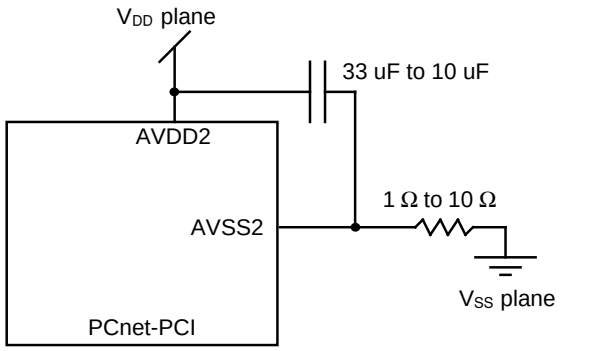
Decoupling/Bypass Capacitors: Adequate decoupling of the power and ground pins and planes is required by all PCnet-PCI designs. This includes both low-frequency bulk capacitors and high frequency capacitors. It is recommended that **at least one** low-frequency bulk (e.g. 22 µF) decoupling capacitor be used in the area of the PCnet-PCI device. The bulk capacitor(s) should be connected directly to the power and ground planes. In addition, **at least 8** high frequency decoupling capacitors (e.g. 0.1 µF multilayer ceramic capacitors) should be used around the periphery of the PCnet-PCI device to prevent power and ground bounce from affecting device operation. To reduce the inductance between the power and ground pins and the capacitors, the pins should be connected directly to the capacitors, rather than through the planes to the capacitors. The suggested connection scheme for the capacitors is shown in the figure below. Note also that the traces connecting these pins to the capacitors should be as wide as possible to reduce inductance (15 mils is desirable).



The most critical pins in the layout of a PCnet-PCI design are the 4 analog power and 2 analog ground pins, AVDD[1–4] and AVSS[1–2], respectively. All of these pins are located in one corner of the device, the "analog corner." Specific functions and layout requirements of the analog power and ground pins are given below.

AVSS1 and AVDD3: These pins provide the power and ground for the Twisted Pair and AUI drivers. In addition AVSS1 serves as the ground for the logic interfaces in the 20 MHz Crystal Oscillator. Hence, these pins can be very noisy. A dedicated 0.1 µF capacitor between these pins is recommended.

AVSS2 and AVDD2: These pins are the **most critical** pins on the PCnet-PCI device because they provide the power and ground for the phase-lock loop (PLL) portion of the chip. The voltage-controlled oscillator (VCO) portion of the PLL is sensitive to noise in the 60 kHz – 200 kHz. range. To prevent noise in this frequency range from disrupting the VCO, it is **strongly recommended** that the low-pass filter shown below be implemented on these pins.



18220C-53

To determine the value for the resistor and capacitor, the formula is:

$$R * C \geq 88$$

Where R is in Ohms and C is in microfarads. Some possible combinations are given below. To minimize the voltage drop across the resistor, the R value should not be more than 10 Ω.

R	C
2.7 Ω	33 µF
4.3 Ω	22 µF
6.8 Ω	15 µF
10 Ω	10 µF

AVSS2 and AVDD2/AVDD4: These pins provide power and ground for the AUI and twisted pair receive circuitry. In addition, as mentioned earlier, AVSS2 and AVDD2 provide power and ground for the phase-lock loop portion of the chip. Except for the filter circuit already mentioned, no specific decoupling is necessary on these pins.

AVDD1: AVDD1 provides power for the control and interface logic in the PLL. Ground for this logic is provided by digital ground pins. No specific decoupling is necessary on this pin.

Special Note for Adapter Cards: In adapter card designs, **it is important to utilize all available power and ground pins available on the bus edge connector**. In addition, the connection from the bus edge connector to the power or ground plane should be made through more than one via and with wide traces (15 mils desirable) wherever possible. Following these recommendations results in minimal inductance in the power and ground paths. By minimizing this inductance, ground bounce is minimized.

APPENDIX C

Alternative Method for Initialization

The PCnet-PCI controller may be initialized by performing I/O writes only. That is, data can be written directly to the appropriate control and status registers (CSR instead of reading from the initialization Block in memory).

The registers that must be written are shown in the table below. These register writes are followed by writing the START bit in CSR0.

Control and Status Register	Comment
CSR8	LADRF[15:0]
CSR9	LADRF[31:16]
CSR10	LADRF[47:32]
CSR11	LADRF[63:48]
CSR12	PADR[15:0]
CSR13	PADR[31:16]
CSR14	PADR[47:32]
CSR15	Mode
CSR24-25	BADR
CSR30-31	BADX
CSR47	POLLINT
CSR76	RCVRL
CSR78	XMTRL

***Note**:*

The INIT bit must not be set or the initialization block will be accessed instead.

Look-Ahead Packet Processing (LAPP) Concept

Introduction of the LAPP Concept

A driver for the PCnet-PCI controller would normally require that the CPU copy receive frame data from the controllers buffer space to the applications buffer space after the entire frame has been received by the controller. For applications that use a ping-pong windowing style, the traffic on the network will be halted until the current frame has been completely processed by the entire application stack. This means that the time between last byte of a receive frame arriving at the clients Ethernet controller and the clients transmission of the first byte of the next outgoing frame will be separated by:

1. The time that it takes the clients CPUs interrupt procedure to pass software control from the current task to the driver
2. plus the time that it takes the client driver to pass the header data to the application and request an application buffer
3. plus the time that it takes the application to generate the buffer pointer and then return the buffer pointer to the driver
4. plus the time that it takes the client driver to transfer all of the frame data from the controllers buffer space into the applications buffer space and then call the application again to process the complete frame
5. plus the time that it takes the application to process the frame and generate the next outgoing frame
6. plus the time that it takes the client driver to set up the descriptor for the controller and then write a TDMD bit to CSR0

The sum of these times can often be about the same as the time taken to actually transmit the frames on the wire, thereby yielding a network utilization rate of less than 50%.

An important thing to note is that the PCnet-PCI controllers data transfers to its buffer space are such that the system bus is needed by the PCnet-PCI controller for approximately 4% of the time. This leaves 96% of the system bus bandwidth for the CPU to perform some of the inter-frame operations in advance of the completion of network receive activity, if possible. The question then becomes: how much of the tasks that need to be performed between reception of a frame and transmission of the next frame can be performed before the reception of the frame actually ends at the network, and how can the CPU be instructed to perform these tasks during the network reception time?

The answer depends upon exactly what is happening in the driver and application code, but the steps that can be performed at the same time as the receive data are arriving include as much as the first 3 steps and part of the 4th step shown in the sequence above. By performing these steps before the entire frame has arrived, the frame throughput can be substantially increased.

A good increase in performance can be expected when the first 3 steps are performed before the end of the network receive operation. A much more significant performance increase could be realized if the PCnet-PCI controller could place the frame data directly into the applications buffer space; (i.e., eliminate the need for step 4.) In order to make this work, it is necessary that the application buffer pointer be determined before the frame has completely arrived, then the buffer pointer in the next descriptor for the receive frame would need to be modified in order to direct the PCnet-PCI controller to write directly to the application buffer. More details on this operation will be given later.

An alternative modification to the existing system can gain a smaller, but still significant improvement in performance. This alternative leaves step 4 unchanged in that the CPU is still required to perform the copy operation, but it allows a large portion of the copy operation to be done before the frame has been completely received by the controller; i.e., the CPU can perform the copy operation of the receive data from the PCnet-PCI controllers buffer space into the application buffer space before the frame data has completely arrived from the network. This allows the copy operation of step 4 to be performed concurrently with the arrival of network data, rather than sequentially, following the end of network receive activity.

Outline of the LAPP Flow

This section gives a suggested outline for a driver that utilizes the LAPP feature of the PCnet-PCI controller.

***Note**: The labels in the following text are used as references in the timeline diagram that follows.*

SETUP:

The driver should set up descriptors in groups of 3, with the OWN and STP bits of each set of three descriptors to read as follows: 11b, 10b, 00b.

An option bit (LAPPEN) exists in CSR3, bit position 5; the software should set this bit; When set, the LAPPEN bit directs the PCnet-PCI controller to generate an INTERRUPT when STP has been written to a receive descriptor by the PCnet-PCI controller.

FLOW:

The PCnet-PCI controller polls the current receive descriptor at some point in time before a message arrives. The PCnet-PCI controller determines that this receive buffer is OWNed by the PCnet-PCI controller and it stores the descriptor information to be used when a message does arrive.

N0: Frame preamble appears on the wire, followed by SFD and destination address.

N1: The 64th byte of frame data arrives from the wire. This causes the PCnet-PCI controller to begin frame data DMA operations to the first buffer.

C0: When the 64th byte of the message arrives, the PCnet-PCI controller performs a lookahead operation to the next receive descriptor. This descriptor should be owned by the PCnet-PCI controller .

C1: The PCnet-PCI controller intermittently requests the bus to transfer frame data to the first buffer as it arrives on the wire.

S1: The driver remains idle.

C2: When the PCnet-PCI controller has completely filled the first buffer, it writes status to the first descriptor.

C3: When the first descriptor for the frame has been written, changing ownership from the PCnet-PCI controller to the CPU, the PCnet-PCI controller will generate an SRP INTERRUPT. (This interrupt appears as a RINT interrupt in CSR0.)

S1: The SRP INTERRUPT causes the CPU to switch tasks to allow the PCnet-PCI controllers driver to run.

C4: During the CPU interrupt-generated task switching, the PCnet-PCI controller is performing a lookahead operation to the third descriptor. At this point in time, the third descriptor is owned by the CPU.

***Note**: Even though the third buffer is not owned by the PCnet-PCI controller, existing AMD Ethernet controllers will continue to perform data DMA into the buffer space that the controller already owns (i.e., buffer number 2). The controller does not know if buffer space in buffer number 2 will be sufficient or not, for this frame, but it has no way to tell except by trying to move the entire message into that space. Only when the message does not fit will it signal a buffer error condition – there is no need to panic at the point that it discovers that it does not yet own descriptor number 3.*

S2: The first task of the drivers interrupt service routine is to collect the header information from the PCnet-PCI controllers first buffer and pass it to the application.

S3: The application will return an application buffer pointer to the driver. The driver will add an offset to the application data buffer pointer, since the PCnet-PCI controller will be placing the first portion of the message into the first and second buffers. (The modified application data buffer pointer will only be directly used by the PCnet-PCI controller when it reaches the third buffer.) The driver will place the modified data buffer pointer into the final descriptor of the group (#3) and will grant ownership of this descriptor to the PCnet-PCI controller.

C5: Interleaved with S2, S3 and S4 driver activity, the PCnet-PCI controller will write frame data to buffer number 2.

S4: The driver will next proceed to copy the contents of the PCnet-PCI controllers first buffer to the beginning of the application space. This copy will be to the exact (unmodified) buffer pointer that was passed by the application.

S5: After copying all of the data from the first buffer into the beginning of the application data buffer, the driver will begin to poll the ownership bit of the second descriptor. The driver is waiting for the PCnet-PCI controller to finish filling the second buffer.

C6: At this point, knowing that it had not previously owned the third descriptor, and knowing that the current message has not ended (there is more data in the FIFO), the PCnet-PCI controller will make a last ditch lookahead to the final (third) descriptor. This time, the ownership will be TRUE (i.e. the descriptor belongs to the controller), because the driver wrote the application pointer into this descriptor and then changed the ownership to give the descriptor to the PCnet-PCI controller back at S3. Note that if steps S1, S2 and S3 have not completed at this time, a BUFF error will result.

C7: After filling the second buffer and performing the last chance lookahead to the next descriptor, the PCnet-PCI controller will write the status and change the ownership bit of descriptor number 2.

S6: After the ownership of descriptor number 2 has been changed by the PCnet-PCI controller, the next driver poll of the 2nd descriptor will show ownership granted to the CPU. The driver now copies the data from buffer number 2 into the middle section of the application buffer space. This operation is interleaved with the C7 and C8 operations.

C8: The PCnet-PCI controller will perform data DMA to the last buffer, whose pointer is pointing to application space. Data entering the last buffer will not need the infamous double copy that is required by existing drivers, since it is being placed directly into the application buffer space.

N2: The message on the wire ends.

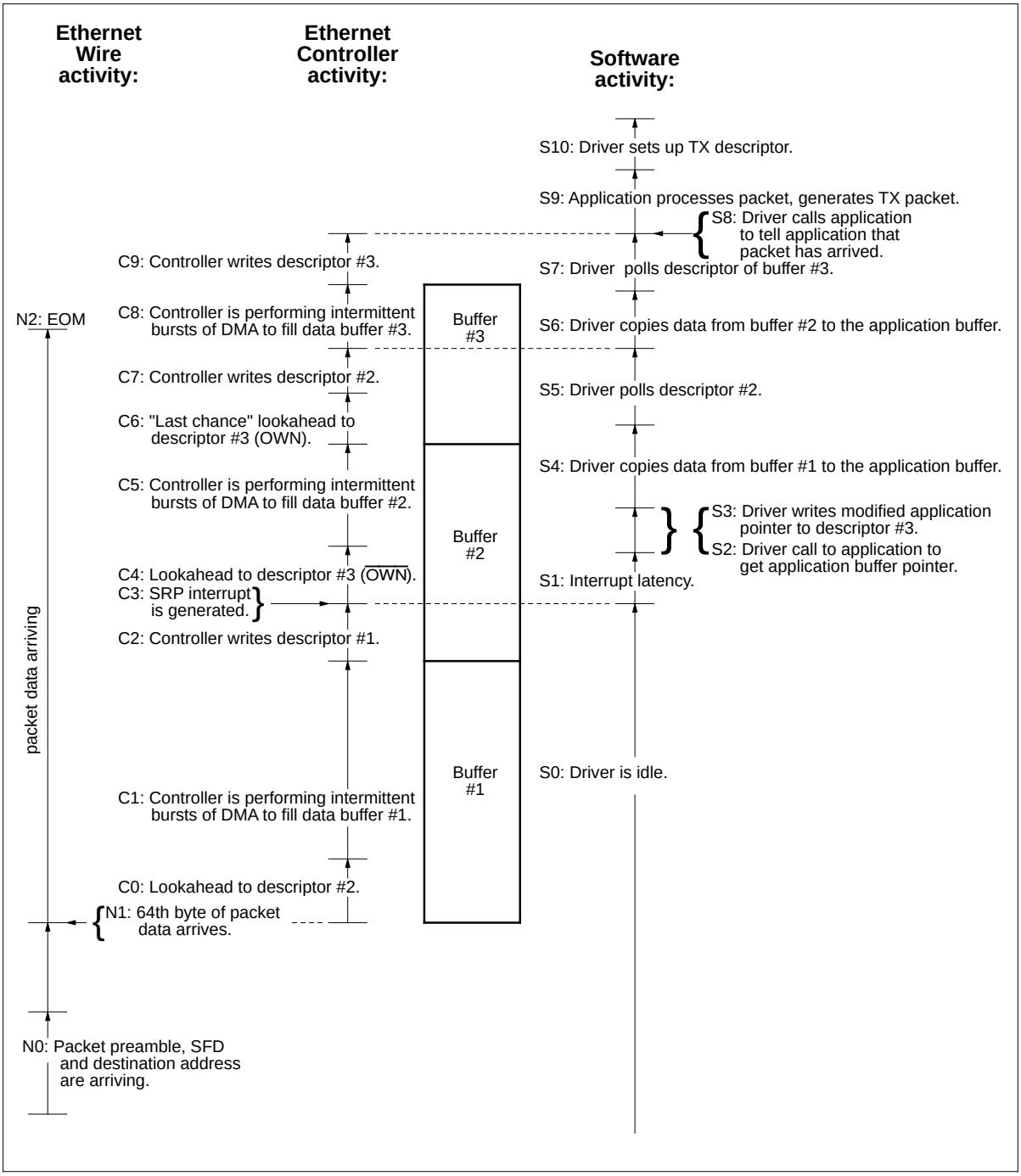
S7: When the driver completes the copy of buffer number 2 data to the application buffer space, it begins polling descriptor number 3.

C9: When the PCnet-PCI controller has finished all data DMA operations, it writes status and changes ownership of descriptor number 3.

S8: The driver sees that the ownership of descriptor number 3 has changed, and it calls the application to tell the application that a frame has arrived.

S9: The application processes the received frame and generates the next TX frame, placing it into a TX buffer.

S10: The driver sets up the TX descriptor for the PCnet-PCI controller.



18220A-55

Figure D1. LAPP Timeline

LAPP Software Requirements:

Software needs to set up a receive ring with descriptors formed into groups of 3. The first descriptor of each group should have OWN = 1 and STP = 1, the second descriptor of each group should have OWN = 1 and STP = 0. The third descriptor of each group should have OWN = 0 and STP = 0. The size of the first buffer (as indicated in the first descriptor), should be at least equal to the largest expected header size; however, for maximum efficiency of CPU utilization, the first buffer size should be larger than the header size. It should be equal to the expected number of message bytes, minus the time needed for Interrupt latency and minus the applica-

tion call latency, minus the time needed for the driver to write to the third descriptor, minus the time needed for the driver to copy data from buffer #1 to the application buffer space, and minus the time needed for the driver to copy data from buffer #2 to the application buffer space. Note that the time needed for the copies performed by the driver depends upon the sizes of the 2nd and 3rd buffers, and that the sizes of the second and third buffers need to be set according to the time needed for the data copy operations! This means that an iterative self-adjusting mechanism needs to be placed into the software to determine the correct buffer sizing for optimal operation. Fixed values for buffer sizes may be used; in such a case, the LAPP method will still provide a significant performance increase, but the performance increase will not be maximized.

The following diagram illustrates this setup for a receive ring size of 9:

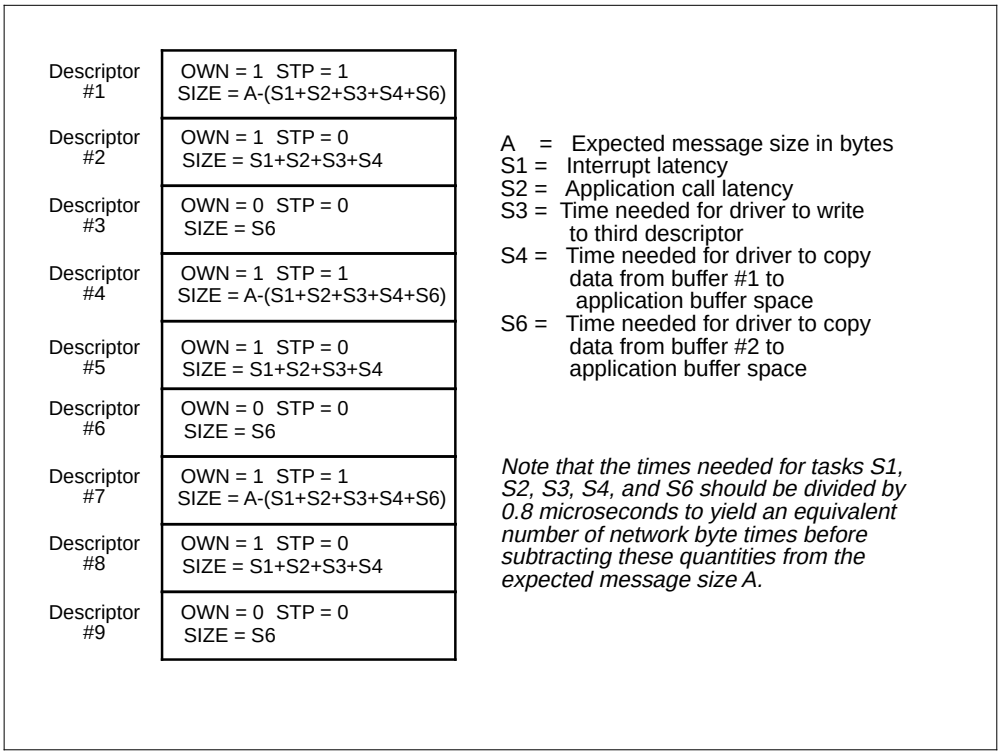


Figure D2. LAPP 3 Buffer Grouping

LAPP Rules for Parsing of Descriptors

When using the LAPP method, software must use a modified form of descriptor *parsing* as follows:

- Software will examine OWN and STP to determine where a RCV frame begins. RCV frames will only begin in buffers that have OWN = 0 and STP = 1.
- Software shall assume that a frame continues until it finds either ENP = 1 or ERR= 1.
- Software must discard all descriptors with OWN = 0 and STP = 0 and move to the next descriptor when searching for the beginning of a new frame; ENP and ERR should be ignored by software during this search.
- Software cannot change an STP value in the receive descriptor ring after the initial setup of the ring is complete, even if software has ownership of the STP descriptor unless the previous STP descriptor in the ring is also OWNED by the software.

When LAPPEN = 1, then hardware will use a modified form of descriptor *parsing* as follows:

- The controller will examine OWN and STP to determine where to begin placing a RCV frame. A new RCV frame will only begin in a buffer that has OWN = 1 and STP = 1.
- The controller will always obey the OWN bit for determining whether or not it may use the next buffer for a chain.
- The controller will always mark the end of a frame with either ENP = 1 or ERR= 1.

The controller will discard all descriptors with OWN = 1 and STP = 0 and move to the next descriptor when searching for a place to begin a new frame. It discards these descriptors by simply changing the ownership bit from OWN=1 to OWN = 0. Such a descriptor is unused for receive purposes by the controller, and the driver must recognize this. (The driver will recognize this if it follows the software rules).

The controller will ignore all descriptors with OWN = 0 and STP = 0 and move to the next descriptor when searching for a place to begin a new frame. In other words, the controller is allowed to skip entries in the ring that it does not own, but only when it is looking for a place to begin a new frame.

Some Examples of LAPP Descriptor Interaction

Choose an expected frame size of 1060 bytes. Choose buffer sizes of 800, 200 and 200 bytes.

- Assume that a 1060 byte frame arrives correctly, and that the timing of the early interrupt and the software is smooth. The descriptors will have changed from:

Descriptor Number	Before the Frame Arrives			After the Frame has Arrived			Comments
	OWN	STP	ENP	OWN	STP	ENP†	(after frame arrival)
1	1	1	X	0	1	0	bytes 1–800
2	1	0	X	0	0	0	bytes 801–1000
3	0	0	X	0	0	1	bytes 1001–1060
4	1	1	X	1	1	X	controller's current location
5	1	0	X	1	0	X	not yet used
6	0	0	X	0	0	X	not yet used
etc.	1	1	X	1	1	X	not yet used

† *ENP or ERR*

- Assume that instead of the expected 1060 byte frame, a 900 byte frame arrives, either because there was an error in the network, or because this is the last frame in a file transmission sequence.

Descriptor Number	Before the Frame Arrives			After the Frame has Arrived			Comments
	OWN	STP	ENP	OWN	STP	ENP†	(after frame arrival)
1	1	1	X	0	1	0	bytes 1–800
2	1	0	X	0	0	1	bytes 801–900
3	0	0	X	0	0	?*	discarded buffer
4	1	1	X	1	1	X	controller's current location
5	1	0	X	1	0	X	not yet used
6	0	0	X	0	0	X	not yet used
etc.	1	1	X	1	1	X	not yet used

† *ENP or ERR*

Note that the PCnet-PCI controller might write a ZERO to ENP location in the 3rd descriptor. Here are the two possibilities:

1. If the controller finishes the data transfers into buffer number 2 after the driver writes the applications modified buffer pointer into the third descriptor, then the controller will write a ZERO to ENP for this buffer and will write a ZERO to OWN and STP.
2. If the controller finishes the data transfers into buffer number 2 before the driver writes the applications modified buffer pointer into the third descriptor, then the controller will complete the frame in buffer number two and then skip the then unowned third buffer. In this case, the PCnet-PCI controller will not have had the opportunity to RESET the ENP bit in this descriptor, and it is possible that the software left this bit as ENP=1 from the last time through the ring. Therefore, the software must treat the location as a don't care; The rule is, after finding ENP=1 (or ERR=1) in descriptor number 2, the software must ignore ENP bits until it finds the next STP=1.

   Assume that instead of the expected 1060 byte frame, a 100 byte frame arrives, because there was an error in the network, or because this is the last frame in a file transmission sequence, or perhaps because it is an acknowledge frame.

* Same as note in case 2 above, except that in this case, it is very unlikely that the driver can respond to the interrupt and get the pointer from the application before the PCnet-PCI controller has completed its poll of the next descriptors. This means that for almost all occurrences of this case, the PCnet-PCI controller will not find the OWN bit set for this descriptor and therefore, the ENP bit will almost always contain the old value, since the PCnet-PCI controller will not have had an opportunity to modify it.

** Note that even though the PCnet-PCI controller will write a ZERO to this ENP location, the software *should* treat the location as a don't care, since after finding the ENP=1 in descriptor number 2, the software should ignore ENP bits until it finds the next STP=1.

Descriptor Number	Before the Frame Arrives			After the Frame has Arrived			Comments
	OWN	STP	ENP	OWN	STP	ENP†	(after frame arrival)
1	1	1	X	0	1	1	bytes 1–100
2	1	0	X	0	0	0**	discarded buffer
3	0	0	X	0	0	?*	discarded buffer
4	1	1	X	1	1	X	controller's current location
5	1	0	X	1	0	X	not yet used
6	0	0	X	0	0	X	not yet used
etc.	1	1	X	1	1	X	not yet used

† *ENP or ERR*

Buffer Size Tuning

For maximum performance, buffer sizes should be adjusted depending upon the expected frame size and the values of the interrupt latency and application call latency. The best driver code will minimize the CPU utilization while also minimizing the latency from frame end on the network to frame sent to application from driver (frame latency). These objectives are aimed at increasing throughput on the network while decreasing CPU utilization.

Note that the buffer sizes in the ring may be altered at any time that the CPU has ownership of the corresponding descriptor. The best choice for buffer sizes will maximize the time that the driver is swapped out, while minimizing the time from the last byte written by the PCnet-PCI controller to the time that the data is passed from the driver to the application. In the diagram, this corresponds to maximizing S0, while minimizing the time between C9 and S8. (The timeline happens to show a minimal time from C9 to S8.)

Note that by increasing the size of buffer number 1, we increase the value of S0. However, when we increase the size of buffer number 1, we also increase the value of S4. If the size of buffer number 1 is too large, then the driver will not have enough time to perform tasks S2, S3, S4, S5 and S6. The result is that there will be delay from the execution of task C9 until the execution of task S8. A *perfectly timed system will have the values for S5 and S7 at a minimum.*

An average increase in performance can be achieved if the general guidelines of buffer sizes in figure 2 is followed. However, as was noted earlier, the correct sizing for buffers will depend upon the expected message size. There are two problems with relating expected message size with the correct buffer sizing:

1. Message sizes cannot always be accurately predicted, since a single application may expect different message sizes at different times, therefore, the buffer sizes chosen will not always maximize throughput.
2. Within a single application, message sizes might be somewhat predictable, but when the same driver is to be shared with multiple applications, there may not be a common predictable message size.

Additional problems occur when trying to define the correct sizing because the correct size also depends upon the interrupt latency, which may vary from system to system, depending upon both the hardware and the software installed in each system.

In order to deal with the unpredictable nature of the message size, the driver can implement a self tuning mechanism that examines the amount of time spent in tasks S5 and S7 as such: while the driver is polling for each descriptor, it could count the number of poll operations performed and then adjust the number 1 buffer size to a larger value, by adding "t" bytes to the buffer count, if the number of poll operations was greater than "x". If fewer than "x" poll operations were needed for each of S5 and S7, then the software should adjust the buffer size to a smaller value by, subtracting "y" bytes from the buffer count. Experiments with such a tuning mechanism must be performed to determine the best values for "X" and "y".

Note whenever the size of buffer number 1 is adjusted, buffer sizes for buffer number 2 and buffer 3 should also be adjusted.

In some systems, the typical mix of receive frames on a network for a client application consists mostly of large data frames, with very few small frames. In this case, for maximum efficiency of buffer sizing, when a frame arrives under a certain size limit, the driver should not adjust the buffer sizes in response to the short frame.

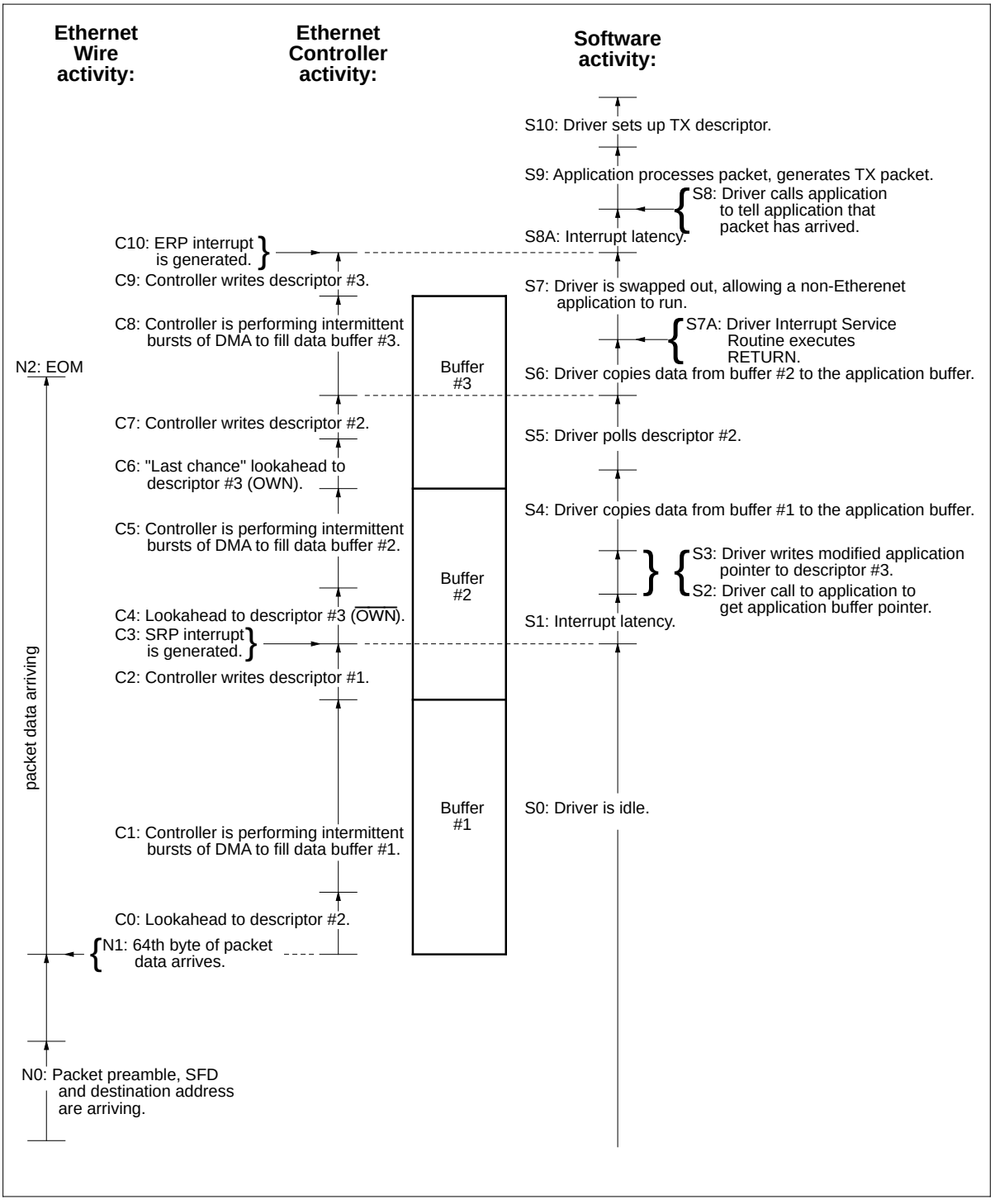
An Alternative LAPP Flow – the TWO Interrupt Method

An alternative to the above suggested flow is to use two interrupts, one at the start of the receive frame and the other at the end of the receive frame, instead of just looking for the SRP interrupt as was described above. This alternative attempts to reduce the amount of time that the software wastes while polling for descriptor own bits. This time would then be available for other CPU tasks. It also minimizes the amount of time the CPU needs for data copying. This savings can be applied to other CPU tasks.

The time from the end of frame arrival on the wire to delivery of the frame to the application is labeled as frame latency. For the one-interrupt method, frame latency is minimized, while CPU utilization increases. For the two-interrupt method, frame latency becomes greater, while CPU utilization decreases.

Note that some of the CPU time that can be applied to non-Ethernet tasks is used for task switching in the CPU. One task switch is required to swap a non-Ethernet task into the CPU (after S7A) and a second task switch is needed to swap the Ethernet driver back in again (at S8A). If the time needed to perform these task switches exceeds the time saved by not polling descriptors, then there is a net loss in performance with this method. Therefore, the LAPP method implemented should be carefully chosen.

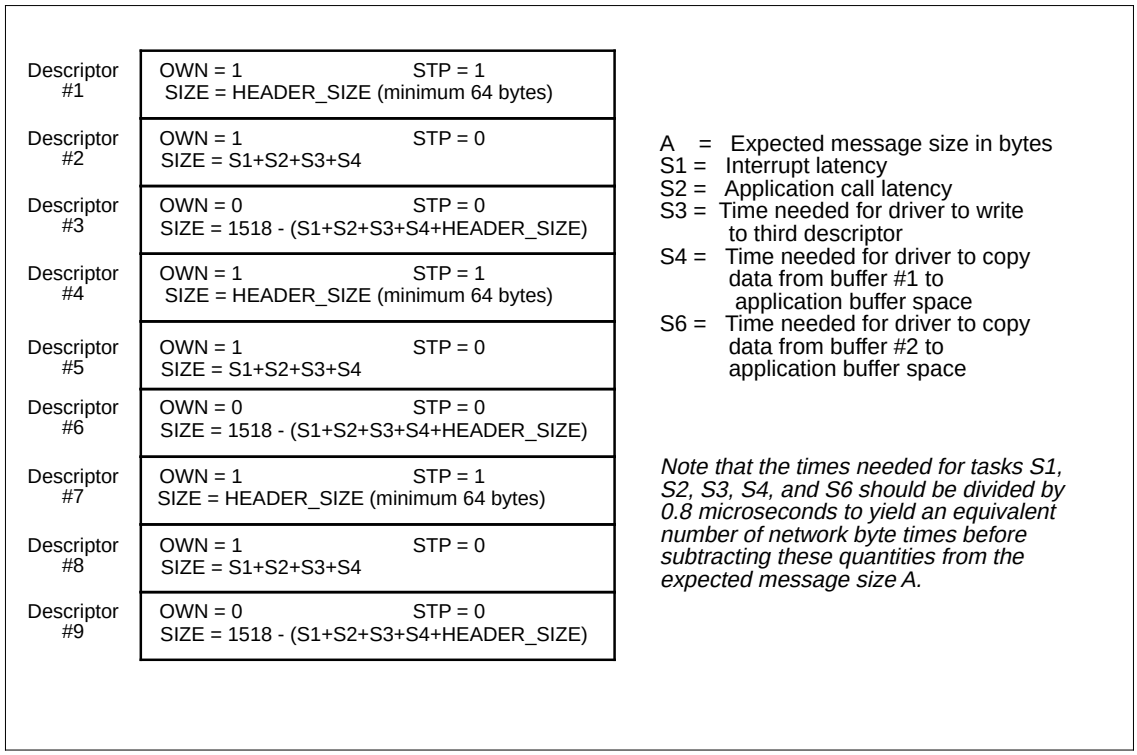
Figure D3 shows the event flow for the two-interrupt method:



18220A-57

Figure D3. LAPP Timeline for TWO-INTERRUPT Method

Figure D4 shows the buffer sizing for the two-interrupt method. Note that the second buffer size will be about the same for each method.



18220A-58

Figure D4. LAPP 3 Buffer Grouping for TWO-INTERRUPT Method

There is another alternative which is a marriage of the two previous methods. This third possibility would use the buffer sizes set by the two-interrupt method, but would use the polling method of determining frame end. This will give good frame latency but at the price of very high CPU utilization.

And still, there are even more compromise positions that use various fixed buffer sizes and effectively, the flow of the one-interrupt method. All of these compromises will reduce the complexity of the one-interrupt method by removing the heuristic buffer sizing code, but they all become less efficient than heuristic code would allow.

DATA SHEET REVISION SUMMARY

The following list represents the key differences between revision B (May 1994) and revision C (June 1994).

Global Change

DWIO mode cannot be set by reading the EEPROM or writing directly to BCR18.

The Initialization Block must be on a double-word boundary.

Detailed Functions

Page 1-888:

The section on PCnet-PCI controller I/O Resource Mapping is rewritten to reflect changes in methods of setting DWIO mode.

User Accessible Registers

Page 1-955:

CSR1
The description for Initialization Block boundary requirement is rewritten for clarity.

Page 1-967:

CSR58
The description for bit 8 (SSIZE32) is rewritten for clarity.

Page 1-982:

BCR18
The description for bit 7 (DWIO) is rewritten for clarity.

The following list represents the key differences between revision A (October 1993) and revision B (April 1994).

Page 1-987:

BCR20
The description for bit 8 (SSIZE32) is rewritten for clarity.

Global Change

Look-Ahead Packet Processing (LAPP) is the name for the early protocol analysis.

Block Diagram

Page 1-869:

The $\overline{\text{LOCK}}$ pin is changed from bidirectional to unidirectional. It is now an input only. The EESK/$\overline{\text{LED1}}$ pin is now changed to bidirectional for typographical correction.

Connection Diagram

Page 1-877:

Various pin names have been changed for either enhanced clarity or to correct typographical errors.

The pins that are affected are 9, 58, 91, 94, 96, 97, 98, 100, 103, 108, 116, and 127.

Pin Designations

Page 1-879:

Various pin names have been changed for either enhanced clarity or to correct typographical errors.

The pins that are affected are 9, 58, 91, 94, 96, 97, 98, 100, 103, 108, 116, and 127.

Page 1-880:

The $\overline{\text{LOCK}}$ pin is changed from an I/O to an input. No driver type is available.

Page 1-881:

The I_{OH} value for TS3 and TS6 are now –2. Driver type O6 is removed. Driver type O8 is added.

Pin Description

Page 1-882:

Pin descriptions for various pins were rewritten for clarity. The pins are $\overline{\text{GNT}}$, $\overline{\text{LOCK}}$, and $\overline{\text{PERR}}$.

Detailed Functions
Page 1-937:

Table 8

EEPROM Contents—Corrected the Hardware ID (byte address 09h) value to 11h.

Page 1-944:
Figure 30

NAND Tree—Typographical errors were corrected.

User Accessible Regisers

Page 1-956:

CSR3
The bit name for bit 5 is changed to LAPPEN (Look-Ahead Packet Processing ENable).

Page 1-958:

CSR4
The bit name for bit 9 is changed to MFCO (Missed Frame Counter Overflow), and the bit name for bit 8 is changed to MFCOM (Missed Frame Counter Overflow Mask).

Page 1-971:

CSR80—The description for bits 9–8 is rewritten for clarity.

Page 1-974:

CSR89—The upper 12 bits of the PCnet–PCI controller part number contained in bits 11–0 are corrected. The 12 bits now read 0010 0100 0011b.

CSR90—The description for this register is removed, because it is now a reserved register.

Page 1-975:

CSR124—The description for bit 3 is rewritten for clarity.

Page 1-982:

BCR18—The descriptions for bits 5 and 6 are rewritten for clarity.

Page 1-994:

Tabel 18—The table is cleaned up for clarity.

DC Characteristics

Page 1-1001:

V_{OL}
The maximum value is 0.55 V if I_{OL} is 3 or 6 mA. The maximum value is 0.4 V if I_{OL} is 12 mA.

Page 1-1003:

CO
The maximum value is now 10 pF.

Appendices

Appendix A

This section is updated with the latest information.

Appendix B

This section is rewritten for clarity.

Appendix D

This is a new section on the LAPP Concept.

Trademarks

Copyright © 1994 Advanced Micro Devices, Inc. All rights reserved.

AMD and the AMD logo are registered trademarks of Advanced Micro Devices, Inc.

PCnet, HIMIB, MACE, and Integrated Local Area Communications Controller (ILACC) are trademarks of Advanced Micro Devices, Inc.

Microsoft is a registered trademark of Microsoft Corporation.

Product names used in this publication are for identification purposes only and may be trademarks of their respective companies.