Am79C972

PCnet™-*FAST*+
Enhanced 10/100 Mbps PCI Ethernet Controller with OnNow Support

DISTINCTIVE CHARACTERISTICS

- Integrated Fast Ethernet controller for the Peripheral Component Interconnect (PCI) bus
  - — 32-bit glueless PCI host interface
  - — Supports PCI clock frequency from DC to 33 MHz independent of network clock
  - — Supports network operation with PCI clock from 15 MHz to 33 MHz
  - — High performance bus mastering architecture with integrated Direct Memory Access (DMA) Buffer Management Unit for low CPU and bus utilization
  - — PCI specification revision 2.1 compliant
  - — Supports PCI Subsystem/Subvendor ID/ Vendor ID programming through the EEPROM interface
  - — Supports both PCI 3.3-V and 5.0-V signaling environments
  - — Plug and Play compatible
  - — Supports an unlimited PCI burst length
  - — Big endian and little endian byte alignments supported
  - — Implements optional PCI power management event ($\overline{\text{PME}}$) pin
- Media Independent Interface (MII) for connecting external 10/100 megabit per second (Mbps) transceivers
  - — IEEE 802.3-compliant MII
  - — Intelligent Auto-Poll™ external PHY status monitor and interrupt
  - — Supports both auto-negotiable and non auto-negotiable external PHYs
  - — Supports 10BASE-T, 100BASE-TX/FX, 100BASE-T4, and 100BASE-T2 IEEE 802.3-compliant MII PHYs at full- or half-duplex
- Supports General Purpose Serial Interface (GPSI) with receive frame tagging support for internetworking applications
- Full-duplex operation supported in MII and GPSI ports with independent Transmit (TX) and Receive (RX) channels
- Supports PC97, PC98, and Net PC requirements
  - — Implements full OnNow features including pattern matching and link status wake-up
  - — Implements Magic Packet mode
  - — Magic Packet mode and the physical address loaded from EEPROM at power up without requiring PCI clock
  - — Supports PCI Bus Power Management Interface Specification Version 1.0
  - — Supports Advanced Configuration and Power Interface (ACPI) Specification Version 1.0
  - — Supports Network Device Class Power Management Specification Version 1.0
- Large independent internal TX and RX FIFOs
  - — Programmable FIFO watermarks for both transmit and receive operations
  - — Receive frame queuing for high latency PCI bus host operation
  - — Programmable allocation of buffer space between transmit and receive queues
- Dual-speed CSMA/CD (10 Mbps and 100 Mbps) Media Access Controller (MAC) compliant with IEEE/ANSI 802.3 and Blue Book Ethernet standards
- EEPROM interface supports jumperless design and provides through-chip programming
  - — Supports full programmability of half-/full-duplex operation for external 10/100 Mbps PHYs through EEPROM mapping
  - — Programmable PHY reset output pin capable of resetting external PHY without needing buffering
- Integrated oscillator circuit eliminates need for external crystal
- Extensive programmable LED status support
- Support for operation in industrial temperature range (-40°C to +85°C)

Publication# **21485** Rev: **E** Amendment/**0**
Issue Date: March 2009

Refer to AMD's Website (www.amd.com) for the latest information.

- **Supports up to 1 megabyte (Mbyte) optional Boot PROM or Flash for diskless node application**
- **Look-Ahead Packet Processing (LAPP) data handling technique reduces system overhead by allowing protocol analysis to begin before the end of a receive frame**
- **Programmable Inter Packet Gap (IPG) to address less network aggressive MAC controllers**
- **Offers the Modified Back-Off algorithm to address the *Ethernet Capture Effect***
- **IEEE 1149.1-compliant JTAG Boundary Scan test access port interface and NAND tree test mode for board-level production connectivity test**
- **Software compatible with AMD PCnet Family and LANCE/C-LANCE register and descriptor architecture**
- **Compatible with the existing PCnet Family driver and diagnostic software**
- **Available in 160-pin PQFP and 176-pin TQFP packages**
- **+3.3 V power supply with 5 V tolerant I/Os enables broad system compatibility**
- **Extensive programmable internal/external loopback capabilities**
- **Supports patented External Address Detection Interface (EADI)**

GENERAL DESCRIPTION

The Am79C972 PCnet-*FAST*+ controller is a highly-integrated 32-bit full-duplex, 10/100-Megabit per second (Mbps) Ethernet controller 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 and bridge/router designs. The bus master architecture provides high data throughput and low CPU and system bus utilization. The Am79C972 controller is fabricated with advanced low-power 3.3-V CMOS process to provide low operating current for power sensitive applications.

The Am79C972 PCnet-*FAST*+ controller also has several enhancements over its predecessor, the Am79C971 PCnet-*FAST* device. In addition to integrating the SRAM on chip, it further reduces system implementation cost by the addition of a new EEPROM programmable pin (PHY_RST), an internal oscillator circuit eliminating the need for an external crystal, and the integration of the PAL function needed for Magic Packet application. The PHY_RST pin is implemented to reset the external PHY without increasing the load to the PCI bus and to block $\overline{\text{RST}}$ to the PHY when PG input is LOW.

The 32-bit multiplexed bus interface unit provides a direct interface to the PCI local bus, simplifying the design of an Ethernet node in a PC system. The Am79C972 PCnet-*FAST*+ controller provides the complete interface to an Expansion ROM or Flash device allowing add-on card designs with only a single load per PCI bus interface pin. With its built-in support for both little and big endian byte alignment, this controller also addresses non-PC applications. The Am79C972 controller's advanced CMOS design allows the bus interface to be connected to either a +5-V or a +3.3-V signaling environment. A compliant IEEE 1149.1 JTAG test interface for board-level testing is also provided, as well as a NAND tree test structure for those systems that cannot support the JTAG interface.

The Am79C972 PCnet-*FAST*+ controller is also compliant with the PC97, PC98, and Net PC specifications. It includes the full implementation of the Microsoft OnNow and ACPI specifications, which are backward compatible with the Magic Packet technology, and is compliant with the PCI Bus Power Management Interface Specification by supporting the four power management states (D0, D1, D2, and D3), the optional $\overline{\text{PME}}$ pin, and the necessary configuration and data registers.

The Am79C972 PCnet-*FAST*+ controller is ideally suited for Network PC (Net PC), motherboard, network interface card (NIC), and embedded designs. It is available in a 160-pin Plastic Quad Flat Pack (PQFP) package and also in a 176-pin Thin Quad Flat Pack (TQFP) package for form factor sensitive designs.

The Am79C972 PCnet-*FAST*+ controller is a complete Ethernet node integrated into a single VLSI device. It contains a bus interface unit, a Direct Memory Access (DMA) Buffer Management Unit, an ISO/IEC 8802-3 (IEEE 802.3)-compliant Media Access Controller (MAC), a large Transmit FIFO and a large Receive FIFO, and an IEEE 802.3-compliant MII. Both IEEE 802.3 compliant full-duplex and half-duplex operations are supported on the MII and GPSI interfaces. 10/100 Mbps operation is supported through the MII.

The Am79C972 PCnet-*FAST*+ controller is register compatible with the LANCE™ (Am7990) and C-LANCE™ (Am79C90) Ethernet controllers, and all Ethernet controllers in the PCnet Family *except* ILACC™ (Am79C900), including the PCnet-ISA™ controller (Am79C960), PCnet-ISA+™ (Am79C961),

PCnet-ISA II™ (Am79C961A), PCnet-32™ (Am79C965), PCnet-PCI™ (Am79C970), PCnet-PCI II™ (Am79C970A), and the PCnet-FAST™ (Am79C971). The Buffer Management Unit supports the LANCE and PCnet descriptor software models.

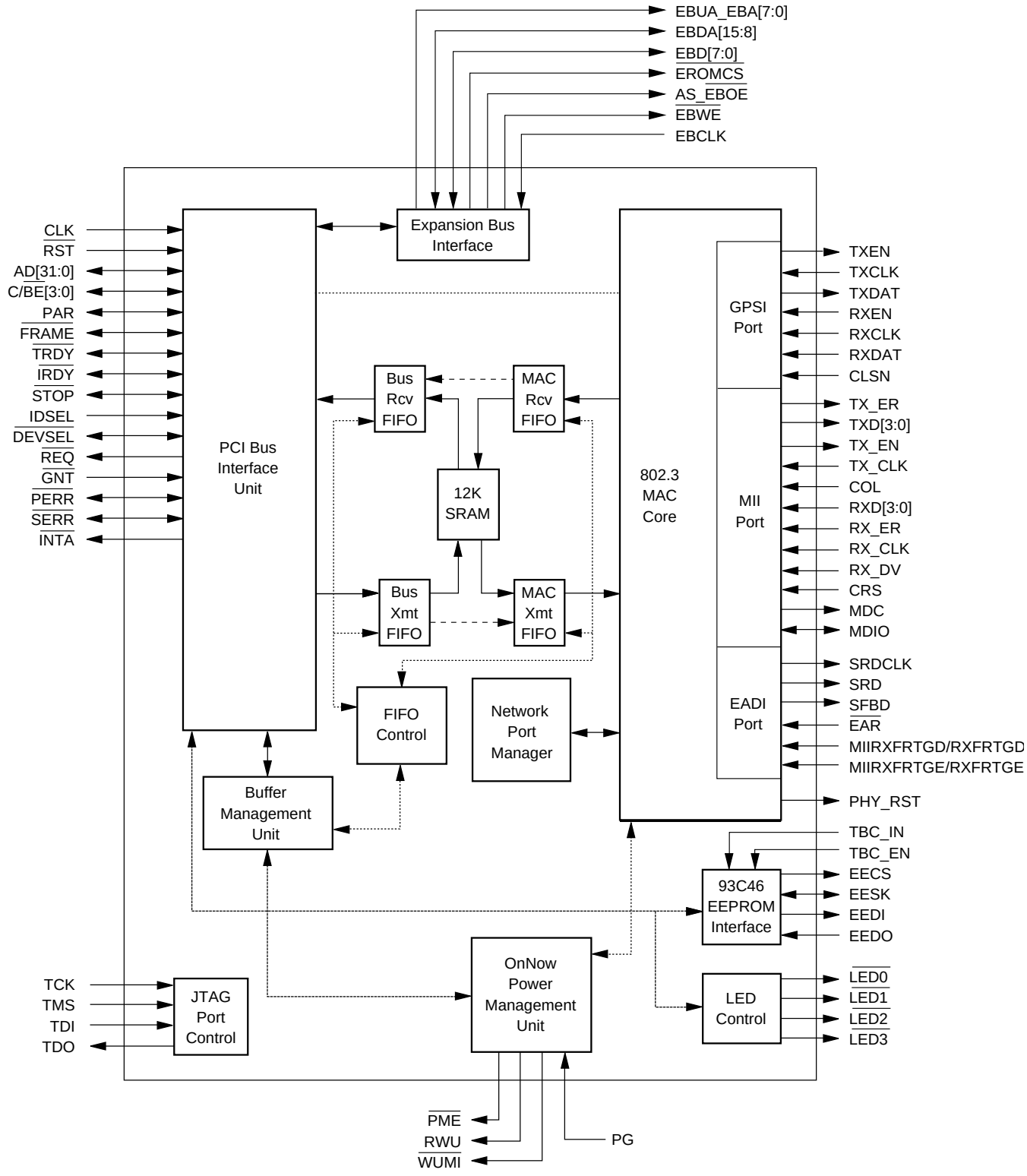
The Am79C972 PCnet-*FAST*+ controller supports auto-configuration in the PCI configuration space. Additional Am79C972 controller configuration parameters, including the unique IEEE physical address, can be read from an external nonvolatile memory (EEPROM) immediately following system reset.

In addition, the device provides programmable on-chip LED drivers for transmit, receive, collision, link integrity, Magic Packet status, activity, address match, full-duplex, or 100 Mbps status. The Am79C972 controller also provides an EADI to allow external hardware address filtering in internetworking applications and a receive frame tagging feature.

The Am79C972 PCnet-*FAST*+ controller contains 12-kilobyte (Kbyte) buffers, the largest of its class of 10/100 Mbps Ethernet controllers. The large internal buffer is programmable between the transmit (TX) and receive (RX) queues for optimal performance.

With the rise of embedded networking applications operating in harsh environments where temperatures may exceed the normal commercial temperature window (0°C to 70°C), an industrial temperature (-40°C to +85°C) version is available in both the 160-pin PQFP and the 176-pin TQFP package. The Am79C972 PCnet-FAST+ 10/100 Mbps Ethernet controller can be designed with the industrial temperature capable Am79C874 NetPHY-1LP 10/100 Mbps Ethernet PHY for a complete and robust Fast Ethernet solution that can withstand extreme temperature environments.

BLOCK DIAGRAM



21485C-1

TABLE OF CONTENTS

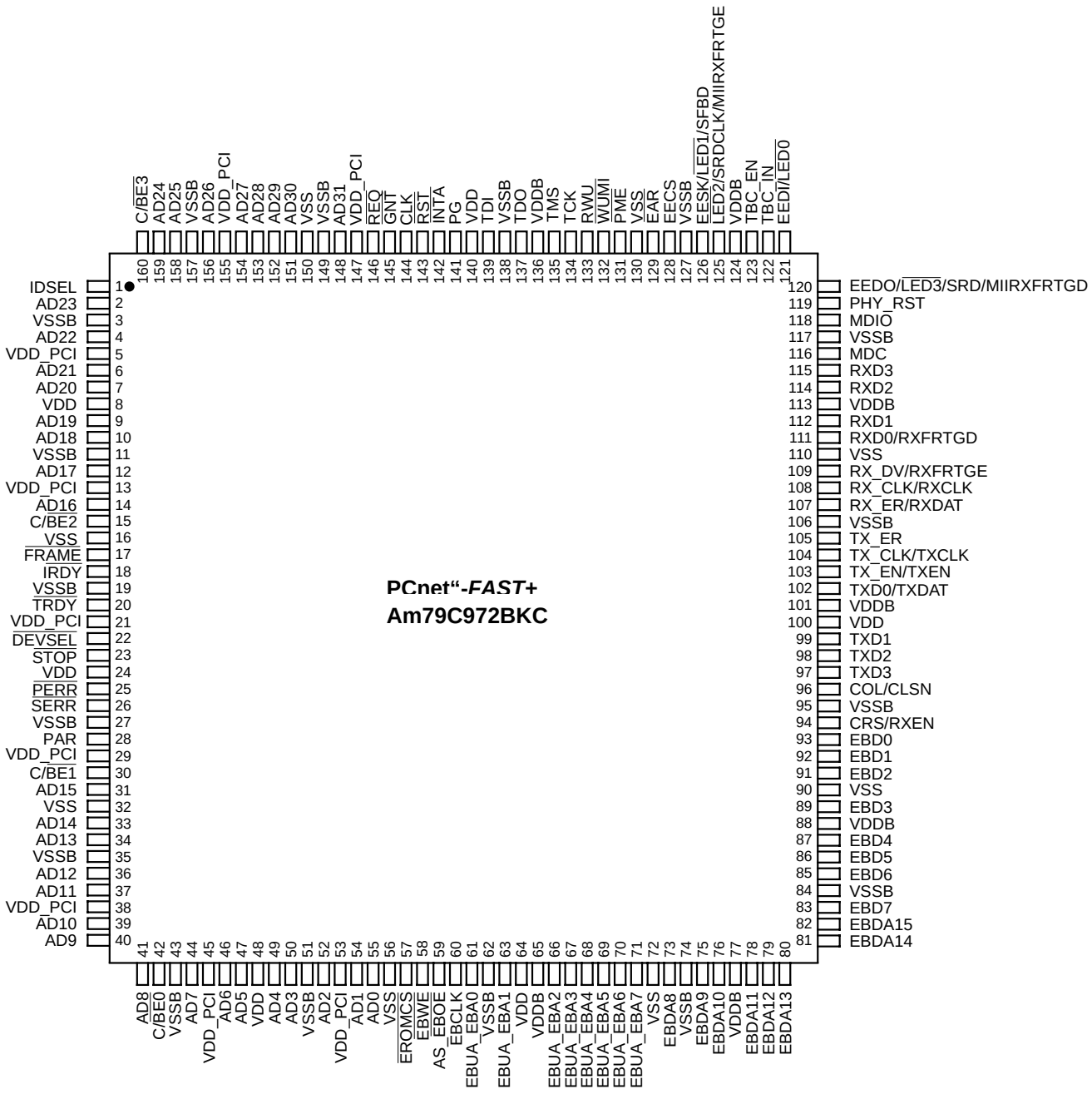
AM79C972 1
DISTINCTIVE CHARACTERISTICS1 **BLOCK DIAGRAM4**
GENERAL DESCRIPTION2
TABLE OF CONTENTS 5
RELATED AMD PRODUCTS 7
CONNECTION DIAGRAM (PQR160)8 **CONNECTION DIAGRAM (PQL176)9**
PIN DESIGNATIONS (PQR160) 10
Listed By Pin Number 10
PIN DESIGNATIONS (PQL176) 11
Listed By Pin Number 11
PIN DESIGNATIONS (PQR160, PQL176) 12
Listed By Group 12
PIN DESIGNATIONS 13
Listed By Group 13
Listed By Driver Type 14
ORDERING INFORMATION 15
Standard Products 15
PIN DESCRIPTIONS 16
PCI Interface 16
Board Interface 18
EEPROM Interface 20
Expansion Bus Interface 20
Media Independent Interface 21
General Purpose Serial Interface 23
External Address Detection Interface 23
IEEE 1149.1 (1990) Test Access Port Interface 24
Power Supply Pins 25
BASIC FUNCTIONS26
System Bus Interface 26
Software Interface 26
Network Interfaces 26
DETAILED FUNCTIONS27
Slave Bus Interface Unit 27
Slave Configuration Transfers 27
Slave I/O Transfers 27
Master Bus Interface Unit 33
Buffer Management Unit 50
Software Interrupt Timer 56
Media Access Control 57
Transmit Operation 60
Receive Operation 62
Loopback Operation 64
General Purpose Serial Interface 65
Media Independent Interface 67
Auto-Negotiation 69
Automatic Network Port Selection 69
External Address Detection Interface 71
Expansion Bus Interface 73
EEPROM Interface 81
LED Support 82
Power Savings Mode 84
Magic Packet Mode 86
IEEE 1149.1 (1990) Test Access Port Interface 88
NAND Tree Testing 89
Reset 91
Software Access 91

USER ACCESSIBLE REGISTERS 95
PCI Configuration Registers 96
RAP Register 105
Control and Status Registers 106
Bus Configuration Registers 139
Initialization Block 171
Receive Descriptors 173
Transmit Descriptors 176
REGISTER SUMMARY 180
PCI Configuration Registers 180
Control and Status Registers 181
Bus Configuration Registers 185
REGISTER PROGRAMMING SUMMARY 186
Am79C972 Programmable Registers 186
ABSOLUTE MAXIMUM RATINGS 190
OPERATING RANGES 190
Commercial (C) Devices 190
Industrial (I) Devices 190
DC CHARACTERISTICS OVER COMMERCIAL AND INDUSTRIAL OPERATING RANGES 190
SWITCHING CHARACTERISTICS: BUS INTERFACE 192
SWITCHING CHARACTERISTICS: MEDIA INDEPENDENT INTERFACE 194
SWITCHING CHARACTERISTICS: GENERAL-PURPOSE SERIAL INTERFACE 195
SWITCHING CHARACTERISTICS: EXTERNAL ADDRESS DETECTION INTERFACE 196
SWITCHING WAVEFORMS 197
Key to Switching Waveforms 197
SWITCHING TEST CIRCUITS 197
SWITCHING WAVEFORMS: SYSTEM BUS INTERFACE 198
SWITCHING WAVEFORMS: EXPANSION BUS INTERFACE 202
SWITCHING WAVEFORMS: MEDIA INDEPENDENT INTERFACE 204
SWITCHING WAVEFORMS: GENERAL-PURPOSE SERIAL INTERFACE 206
SWITCHING WAVEFORMS: EXTERNAL ADDRESS DETECTION INTERFACE 207
SWITCHING WAVEFORMS: RECEIVE FRAME TAG 207
PHYSICAL DIMENSIONS* 208
PQR160 208
Plastic Quad Flat Pack (measured in millimeters) 208
PQL176 209
Thin Quad Flat Pack (measured in millimeters) 209
INTRODUCTION 213
Outline of LAPP Flow 213
LAPP Software Requirements 217
LAPP Rules for Parsing Descriptors 217
Some Examples of LAPP Descriptor Interaction 218
Control Register (Register 0) 223
Status Register (Register 1) 224
Auto-Negotiation Advertisement Register (Register 4) 225
Auto-Negotiation Link Partner Ability Register (Register 5) 226

RELATED AMD PRODUCTS

Part No.	Description
Am79C90	CMOS Local Area Network Controller for Ethernet (C-LANCE)
Am7996	IEEE 802.3/Ethernet/Cheapernet Tap Transceiver
Am79C98	Twisted Pair Ethernet Transceiver (TPEX)
Am79C100	Twisted Pair Ethernet Transceiver Plus (TPEX+)
Am79865	100 Mbps Physical Data Transmitter (PDT)
Am79866A	100 Mbps Physical Data Receiver (PDR)
Am79C871	Quad 100BASE-X Transceiver for Repeater
Am79C940	Media Access Controller for Ethernet (MACE™)
Am79C961A	PCnet-ISA II Single-Chip Full-Duplex Ethernet Controller (with Microsoft® Plug n' Play support)
Am79C965	PCnet-32 Single-Chip 32-Bit Ethernet Controller (for 486 and VL buses)
Am79C970A	PCnet-PCI II Single-Chip Full-Duplex Ethernet Controller for PCI Local Bus
Am79C971	PCnet-*FAST* Single-Chip Full-Duplex 10/100 Ethernet Controller for PCI Local Bus

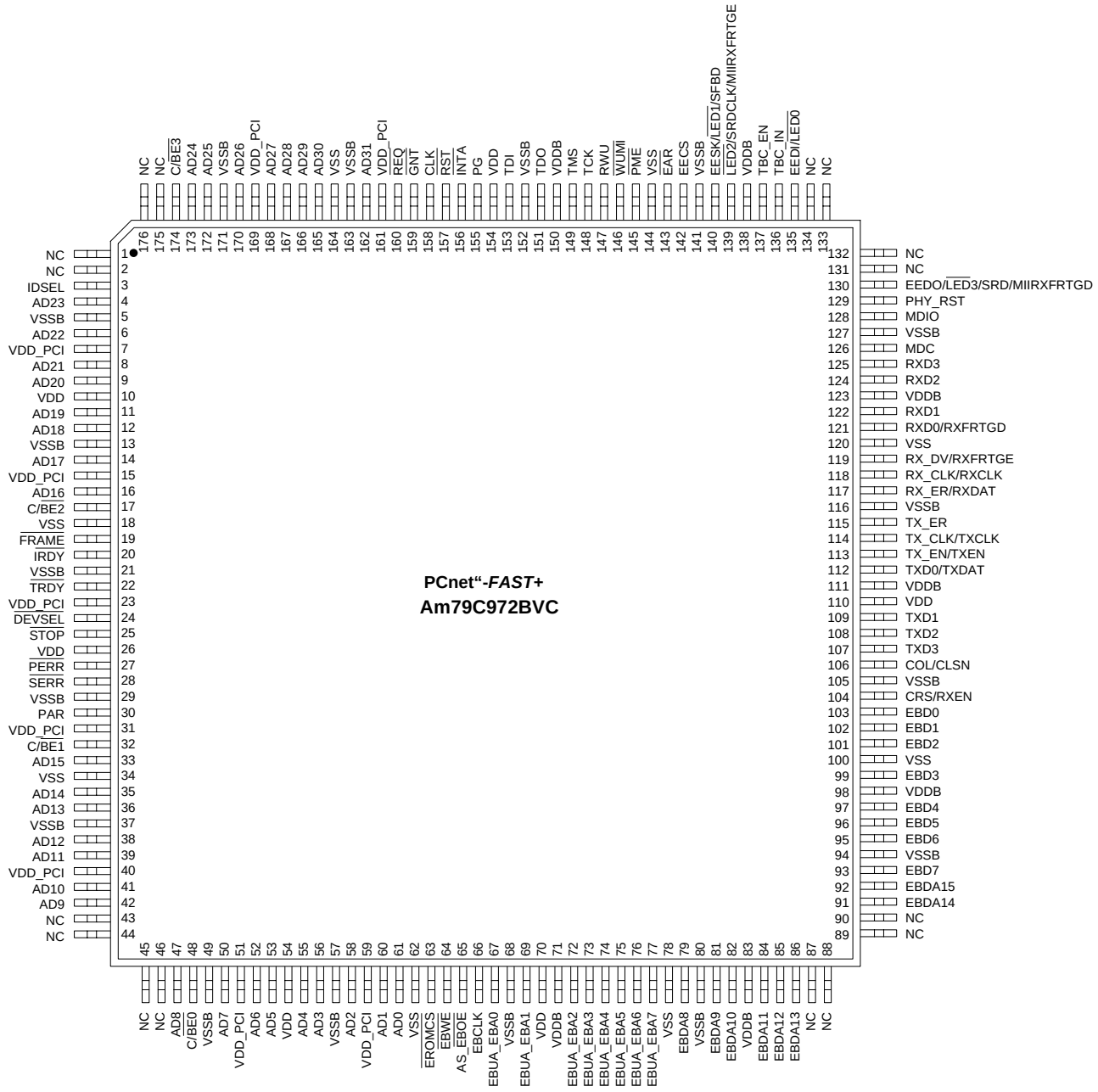
CONNECTION DIAGRAM (PQR160)



21485C-2

Pin 1 is marked for orientation.

CONNECTION DIAGRAM (PQL176)



21485C-3

Pin 1 is marked for orientation.

PIN DESIGNATIONS (PQR160)

Listed By Pin Number

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
1	IDSEL	41	AD8	81	EBDA14	121	EEDI/LED0
2	AD23	42	C/BE0	82	EBDA15	122	TBC_IN
3	VSSB	43	VSSB	83	EBD7	123	TBC_EN
4	AD22	44	AD7	84	VSSB	124	VDDB
5	VDD_PCI	45	VDD_PCI	85	EBD6	125	LED2/SRDCLK/ MIIRXFRTGE
6	AD21	46	AD6	86	EBD5	126	EESK/LED1/SFBD
7	AD20	47	AD5	87	EBD4	127	VSSB
8	VDD	48	VDD	88	VDDB	128	EECS
9	AD19	49	AD4	89	EBD3	129	EAR
10	AD18	50	AD3	90	VSS	130	VSS
11	VSSB	51	VSSB	91	EBD2	131	PME
12	AD17	52	AD2	92	EBD1	132	WUMI
13	VDD_PCI	53	VDD_PCI	93	EBD0	133	RWU
14	AD16	54	AD1	94	CRS/RXEN	134	TCK
15	C/BE2	55	AD0	95	VSSB	135	TMS
16	VSS	56	VSS	96	COL/CLSN	136	VDDB
17	FRAME	57	EROMCS	97	TXD3	137	TDO
18	IRDY	58	EBWE	98	TXD2	138	VSSB
19	VSSB	59	AS_EBOE	99	TXD1	139	TDI
20	TRDY	60	EBCLK	100	VDD	140	VDD
21	VDD_PCI	61	EBUA_EBA0	101	VDDB	141	PG
22	DEVSEL	62	VSSB	102	TXD0/TXDAT	142	INTA
23	STOP	63	EBUA_EBA1	103	TX_EN/TXEN	143	RST
24	VDD	64	VDD	104	TX_CLK/TXCLK	144	CLK
25	PERR	65	VDDB	105	TX_ER	145	GNT
26	SERR	66	EBUA_EBA2	106	VSSB	146	REQ
27	VSSB	67	EBUA_EBA3	107	RX_ER/RXDAT	147	VDD_PCI
28	PAR	68	EBUA_EBA4	108	RX_CLK/RXCLK	148	AD31
29	VDD_PCI	69	EBUA_EBA5	109	RX_DV/RXFRTGE	149	VSSB
30	C/BE1	70	EBUA_EBA6	110	VSS	150	VSS
31	AD15	71	EBUA_EBA7	111	RXD0/RXFRTGD	151	AD30
32	VSS	72	VSS	112	RXD1	152	AD29
33	AD14	73	EBDA8	113	VDDB	153	AD28
34	AD13	74	VSSB	114	RXD2	154	AD27
35	VSSB	75	EBDA9	115	RXD3	155	VDD_PCI
36	AD12	76	EBDA10	116	MDC	156	AD26
37	AD11	77	VDDB	117	VSSB	157	VSSB
38	VDD_PCI	78	EBDA11	118	MDIO	158	AD25
39	AD10	79	EBDA12	119	PHY_RST	159	AD24
40	AD9	80	EBDA13	120	EEDO/LED3/SRD/ MIIRXFRTGD	160	C/BE3

PIN DESIGNATIONS (PQL176)

Listed By Pin Number

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
1	NC	45	NC	89	NC	133	NC
2	NC	46	NC	90	NC	134	NC
3	IDSEL	47	AD8	91	EBDA14	135	EEDI/$\overline{\text{LED0}}$
4	AD23	48	C/$\overline{\text{BE0}}$	92	EBDA15	136	TBC_IN
5	VSSB	49	VSSB	93	EBD7	137	TBC_EN
6	AD22	50	AD7	94	VSSB	138	VDDB
7	VDD_PCI	51	VDD_PCI	95	EBD6	139	$\overline{\text{LED2}}$/SRDCLK/ MIIRXFRTGE
8	AD21	52	AD6	96	EBD5	140	EESK/$\overline{\text{LED1}}$/SFBD
9	AD20	53	AD5	97	EBD4	141	VSSB
10	VDD	54	VDD	98	VDDB	142	EECS
11	AD19	55	AD4	99	EBD3	143	$\overline{\text{EAR}}$
12	AD18	56	AD3	100	VSS	144	VSS
13	VSSB	57	VSSB	101	EBD2	145	$\overline{\text{PME}}$
14	AD17	58	AD2	102	EBD1	146	$\overline{\text{WUMI}}$
15	VDD_PCI	59	VDD_PCI	103	EBD0	147	RWU
16	AD16	60	AD1	104	CRS/RXEN	148	TCK
17	C/$\overline{\text{BE2}}$	61	AD0	105	VSSB	149	TMS
18	VSS	62	VSS	106	COL/CLSN	150	VDDB
19	$\overline{\text{FRAME}}$	63	$\overline{\text{EROMCS}}$	107	TXD3	151	TDO
20	$\overline{\text{IRDY}}$	64	$\overline{\text{EBWE}}$	108	TXD2	152	VSSB
21	VSSB	65	AS_$\overline{\text{EBOE}}$	109	TXD1	153	TDI
22	$\overline{\text{TRDY}}$	66	EBCLK	110	VDD	154	VDD
23	VDD_PCI	67	EBUA_EBA0	111	VDDB	155	PG
24	$\overline{\text{DEVSEL}}$	68	VSSB	112	TXD0/TXDAT	156	$\overline{\text{INTA}}$
25	$\overline{\text{STOP}}$	69	EBUA_EBA1	113	TX_EN/TXEN	157	$\overline{\text{RST}}$
26	VDD	70	VDD	114	TX_CLK/TXCLK	158	CLK
27	$\overline{\text{PERR}}$	71	VDDB	115	TX_ER	159	$\overline{\text{GNT}}$
28	$\overline{\text{SERR}}$	72	EBUA_EBA2	116	VSSB	160	$\overline{\text{REQ}}$
29	VSSB	73	EBUA_EBA3	117	RX_ER/RXDAT	161	VDD_PCI
30	PAR	74	EBUA_EBA4	118	RX_CLK/RXCLK	162	AD31
31	VDD_PCI	75	EBUA_EBA5	119	RX_DV/RXFRTGE	163	VSSB
32	C/$\overline{\text{BE1}}$	76	EBUA_EBA6	120	VSS	164	VSS
33	AD15	77	EBUA_EBA7	121	RXD0/RXFRTGD	165	AD30
34	VSS	78	VSS	122	RXD1	166	AD29
35	AD14	79	EBDA8	123	VDDB	167	AD28
36	AD13	80	VSSB	124	RXD2	168	AD27
37	VSSB	81	EBDA9	125	RXD3	169	VDD_PCI
38	AD12	82	EBDA10	126	MDC	170	AD26
39	AD11	83	VDDB	127	VSSB	171	VSSB
40	VDD_PCI	84	EBDA11	128	MDIO	172	AD25
41	AD10	85	EBDA12	129	PHY_RST	173	AD24
42	AD9	86	EBDA13	130	EEDO/$\overline{\text{LED3}}$/SRD/ MIIRXFRTGD	174	C/$\overline{\text{BE3}}$
43	NC	87	NC	131	NC	175	NC
44	NC	88	NC	132	NC	176	NC

PIN DESIGNATIONS (PQR160, PQL176)

Listed By Group

Pin Name	Pin Function	Type[1]	No. of Pins
PCI Bus Interface			
AD[31:0]	Address/Data Bus	IO	32
C/$\overline{BE}$[3:0]	Bus Command/Byte Enable	IO	4
CLK	Bus Clock	I	1
$\overline{DEVSEL}$	Device Select	IO	1
$\overline{FRAME}$	Cycle Frame	IO	1
$\overline{GNT}$	Bus Grant	I	1
IDSEL	Initialization Device Select	I	1
$\overline{INTA}$	Interrupt	O	1
$\overline{IRDY}$	Initiator Ready	IO	1
PAR	Parity	IO	1
$\overline{PERR}$	Parity Error	IO	1
$\overline{REQ}$	Bus Request	O	1
$\overline{RST}$	Reset	I	1
$\overline{SERR}$	System Error	IO	1
$\overline{STOP}$	Stop	IO	1
$\overline{TRDY}$	Target Ready	IO	1
Board Interface			
$\overline{LED0}$	LED0	O	1
$\overline{LED1}$	LED1	O	1
$\overline{LED2}$	LED2	O	1
$\overline{LED3}$	LED3	O	1
TBC_IN	Test Pin	I	1
TBC_EN	Test Pin	I	1
PHY_RST	Reset to PHY	O	1
EEPROM Interface			
EECS	Serial EEPROM Chip Select	O	1
EEDI	Serial EEPROM Data In	O	1
EEDO	Serial EEPROM Data Out	I	1
EESK	Serial EEPROM Clock	IO	1
Expansion ROM Interface			
AS_$\overline{EBOE}$	Address Strobe/Expansion Bus Output Enable	O	1
EBCLK	Expansion Bus Clock	I	1
EBD[7:0]	Expansion Bus Data [7:0]	IO	8
EBDA[15:8]	Expansion Bus Data/Address [15:8]	IO	8
EBUA_EBA[7:0]	Expansion Bus Upper Address /Expansion Bus Address [7:0]	O	8
$\overline{EBWE}$	Expansion Bus Write Enable	O	1
$\overline{EROMCS}$	Expansion Bus ROM Chip Select	O	1

Note: *1. Not including test features.*

PIN DESIGNATIONS

Listed By Group

Pin Name	Pin Function	Type[1]	No. of Pins
Media Independent Interface (MII)			
COL	Collision	I	1
CRS	Carrier Sense	I	1
MDC	Management Data Clock	O	1
MDIO	Management Data I/O	IO	1
RX_CLK	Receive Clock	I	1
RXD[3:0]	Receive Data	I	4
RX_DV	Receive Data Valid	I	1
RX_ER	Receive Error	I	1
TX_CLK	Transmit Clock	I	1
TXD[3:0]	Transmit Data	O	4
TX_EN	Transmit Data Enable	O	1
TX_ER	Transmit Error	O	1
General Purpose Serial Interface (GPSI)			
CLSN	Collision	I	1
RXCLK	Receive Clock	I	1
RXDAT	Receive Data	I	1
RXEN	Receive Enable	I	1
TXCLK	Transmit Clock	I	1
TXDAT	Transmit Data	O	1
TXEN	Transmit Enable	O	1
External Address Detection Interface (EADI)			
EAR	External Address Reject Low	I	1
SFBD	Start Frame Byte Delimiter	O	1
SRD	Serial Receive Data	O	1
SRDCLK	Serial Receive Data Clock	O	1
RXFRTGD	Receive Frame Tag Data	I	1
RXFRTGE	Receive Frame Tag Enable	I	1
MIIRXFRTGD	MII Receive Frame Tag Data	I	1
MIIRXFRTGE	MII Receive Frame Tag Enable	I	1
Power Management Interface			
RWU	Remote Wake Up	O	1
PME	Power Management Event	O	1
WUMI	Wake-Up Mode Indication	O	1
PG	Power Good	I	1
IEEE 1149.1 Test Access Port Interface (JTAG)			
TCK	Test Clock	I	1
TDI	Test Data In	I	1
TDO	Test Data Out	O	1
TMS	Test Mode Select	I	1
Power Supplies			
VDD	Digital Power	P	6
VSS	Digital Ground	P	8
VDDB	I/O Buffer Power	P	7
VSSB	I/O Buffer Ground	P	17
VDD_PCI	PCI I/O Buffer Power	P	9

Note: *1. Not including test features.*

Listed By Driver Type

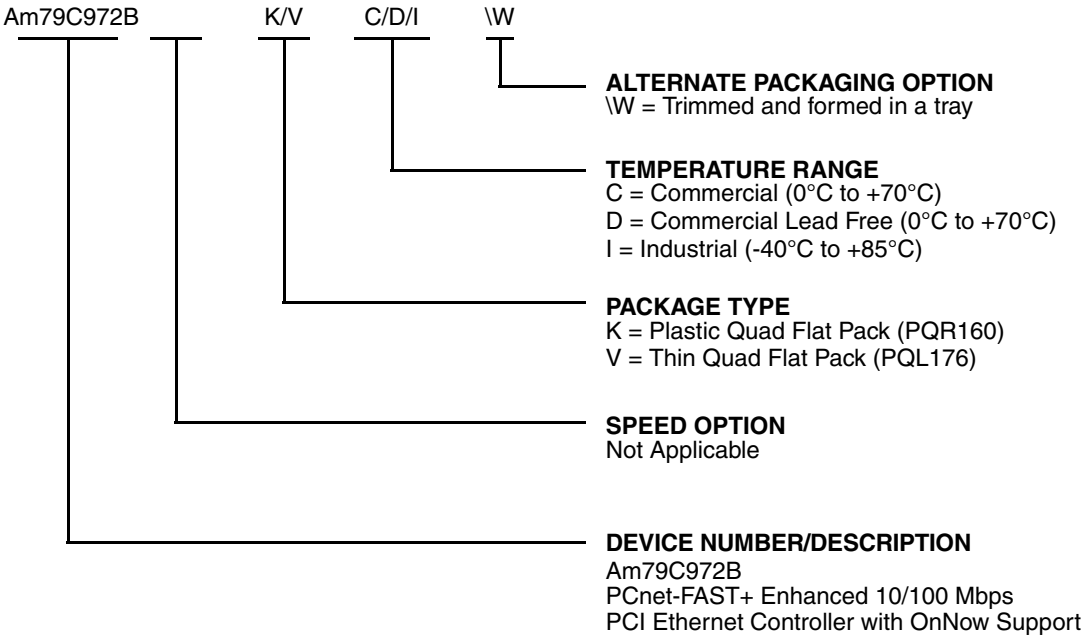
The following table describes the various types of output drivers used in the Am79C972 controller. All I_{OL} and I_{OH} values shown in the table apply to 3.3 V signaling.

A sustained tri-state signal is a low active signal that is driven high for one clock period before it is left floating.

Name	Type	I_{OL} (mA)	I_{OH} (mA)	Load (pF)
LED	LED	12	-0.4	50
OMII1	Totem Pole	4	-4	50
OMII2	Totem Pole	4	-4	390
O6	Totem Pole	6	-0.4	50
OD6	Open Drain	6	NA	50
STS6	Sustained Tri-State	6	-2	50
TS3	Tri-State	3	-2	50
TS6	Tri-State	6	-2	50
TSMII	Tri-State	4	-4	470

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 the elements below.



Valid Combinations

Am79C972B	KC/W VC/W VD/W
Am79C972B	KI/W VI/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 DESCRIPTIONS

PCI Interface

AD[31:0]

Address and Data **Input/Output**

Address and data are multiplexed on the same bus interface pins. During the first clock of a transaction, AD[31:0] contain a physical address (32 bits). During the subsequent clocks, AD[31:0] contain data. Byte ordering is little endian by default. AD[7:0] are defined as the least significant byte (LSB) and AD[31:24] are defined as the most significant byte (MSB). For FIFO data transfers, the Am79C972 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 Am79C972 controller is a bus master, AD[31:2] will address the active Double Word (DWord). The Am79C972 controller always drives AD[1:0] to '00' during the address phase indicating linear burst order. When the Am79C972 controller is not a bus master, the AD[31:0] lines are continuously monitored to determine if an address match exists for slave transfers.

During the data phase of the transaction, AD[31:0] are driven by the Am79C972 controller when performing bus master write and slave read operations. Data on AD[31:0] is latched by the Am79C972 controller when performing bus master read 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**

Bus command and byte enables are multiplexed on the same bus interface 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}$0 applies to byte 0 (AD[7:0]) and C/$\overline{BE}$3 applies to byte 3 (AD[31:24]). The function of the byte enables is independent of the byte ordering mode (BSWP, CSR3, bit 2).

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

CLK

Clock **Input**

This clock is used to drive the system bus interface and the internal buffer management unit. All bus signals are sampled on the rising edge of CLK and all parameters are defined with respect to this edge. The Am79C972 controller normally operates over a frequency range of 10 to 33 MHz on the PCI bus due to networking demands. See the *Frequency Demands for Network Operation* section for details. The Am79C972 controller will support a clock frequency of 0 MHz after certain precautions are taken to ensure data integrity. This clock or a derivation is not used to drive any network functions.

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

$\overline{DEVSEL}$

Device Select **Input/Output**

The Am79C972 controller drives $\overline{DEVSEL}$ when it detects a transaction that selects the device as a target. The device samples $\overline{DEVSEL}$ to detect if a target claims a transaction that the Am79C972 controller has initiated.

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

$\overline{FRAME}$

Cycle Frame **Input/Output**

$\overline{FRAME}$ is driven by the Am79C972 controller when it is the bus master to indicate the beginning and duration of a transaction. $\overline{FRAME}$ is asserted to indicate a bus transaction is beginning. $\overline{FRAME}$ is asserted while data transfers continue. $\overline{FRAME}$ is deasserted before the final data phase of a transaction. When the Am79C972 controller is in slave mode, it samples $\overline{FRAME}$ to determine the address phase of a transaction.

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 Am79C972 controller.

The Am79C972 controller supports bus parking. When the PCI bus is idle and the system arbiter asserts $\overline{GNT}$ without an active $\overline{REQ}$ from the Am79C972 controller, the device will drive the AD[31:0], C/$\overline{BE}$[3:0] 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 Am79C972 controller during configuration read and write transactions.

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

INTA

Interrupt Request **Output**

An attention signal which indicates that one or more of the following status flags is set: EXDINT, IDON, MERR, MISS, MFCO, MPINT, RCVCCO, RINT, SINT, TINT, TXSTRT, UINT, MCCINT, MPDTINT, MAPINT, MREINT, and STINT. Each status flag has either a mask or an enable bit which allows for suppression of INTA assertion. Table 1 shows the flag descriptions. By default INTA is an open-drain output. For applications that need a high-active edge-sensitive interrupt signal, the INTA pin can be configured for this mode by setting INTLEVEL (BCR2, bit 7) to 1.

When RST is active, INTA is the output for NAND tree testing.

IRDY

Initiator Ready **Input/Output**

IRDY indicates the ability of the initiator of the transaction to complete the current data phase. IRDY is used in conjunction with TRDY. Wait states are inserted until both IRDY and TRDY are asserted simultaneously. A data phase is completed on any clock when both IRDY and TRDY are asserted.

When the Am79C972 controller is a bus master, it asserts IRDY during all write data phases to indicate that valid data is present on AD[31:0]. During all read data phases, the device asserts IRDY to indicate that it is ready to accept the data.

When the Am79C972 controller is the target of a transaction, it checks IRDY during all write data phases to determine if valid data is present on AD[31:0]. During all read data phases, the device checks IRDY to determine if the initiator is ready to accept the data.

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

PAR

Parity **Input/Output**

Parity is even parity across AD[31:0] and C/BE[3:0]. When the Am79C972 controller is a bus master, it generates parity during the address and write data phases. It checks parity during read data phases. When the Am79C972 controller operates in slave mode, it checks parity during every address phase. When it is the target of a cycle, it checks parity during write data phases and it generates parity during read data phases.

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

.

Table 1. Interrupt Flags

Name	Description	Mask Bit	Interrupt Bit
EXDINT	Excessive Deferral	CSR5, bit 6	CSR5, bit 7
IDON	Initialization Done	CSR3, bit 8	CSR0, bit 8
MERR	Memory Error	CSR3, bit 11	CSR0, bit 11
MISS	Missed Frame	CSR3, bit 12	CSR0, bit 12
MFCO	Missed Frame Count Overflow	CSR4, bit 8	CSR4, bit 9
MPINT	Magic Packet Interrupt	CSR5, bit 3	CSR5, bit 4
RCVCCO	Receive Collision Count Overflow	CSR4, bit 4	CSR4, bit 5
RINT	Receive Interrupt	CSR3, bit 10	CSR0, bit 10
SINT	System Error	CSR5, bit 10	CSR5, bit 11
TINT	Transmit Interrupt	CSR3, bit 9	CSR0, bit 9
TXSTRT	Transmit Start	CSR4, bit 2	CSR4, bit 3
UINT	User Interrupt	CSR4, bit 7	CSR4, bit 6
MCCINT	MII Management Command Complete Interrupt	CSR7, bit 4	CSR7, bit 5
MPDTINT	MII PHY Detect Transition Interrupt	CSR7, bit 0	CSR7, bit 1
MAPINT	MII Auto-Poll Interrupt	CSR7, bit 6	CSR7, bit 7
MREINT	MII Management Frame Read Error Interrupt	CSR7, bit 8	CSR7, bit 9
STINT	Software Timer Interrupt	CSR7, bit 10	CSR7, bit 11

PERR

Parity Error **Input/Output**

During any slave write transaction and any master read transaction, the Am79C972 controller asserts PERR when it detects a data parity error and reporting of the error is enabled by setting PERREN (PCI Command register, bit 6) to 1. During any master write transaction, the Am79C972 controller monitors PERR to see if the target reports a data parity error.

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

REQ

Bus Request **Input/Output**

The Am79C972 controller asserts REQ pin as a signal that it wishes to become a bus master. REQ is driven high when the Am79C972 controller does not request the bus. In Power Management mode, the REQ pin will not be driven.

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

RST

Reset **Input**

When RST is asserted LOW and the PG pin is HIGH, then the Am79C972 controller performs an internal system reset of the type H_RESET (HARDWARE_RESET, see section on RESET). RST must be held for a minimum of 30 clock periods. While in the H_RESET state, the Am79C972 controller will disable or deassert all outputs. RST may be asynchronous to clock when asserted or deasserted.

When the PG pin is LOW, RST disables all of the PCI pins except the PME pin.

When RST is LOW and PG is HIGH, NAND tree testing is enabled.

SERR

System Error **Output**

During any slave transaction, the Am79C972 controller asserts SERR when it detects an address parity error, and reporting of the error is enabled by setting PEREN (PCI Command register, bit 6) and SERREN (PCI Command register, bit 8) to 1.

By default SERR is an open-drain output. For component test, it can be programmed to be an active-high totem-pole output.

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

STOP

Stop **Input/Output**

In slave mode, the Am79C972 controller drives the STOP signal to inform the bus master to stop the current transaction. In bus master mode, the Am79C972 controller checks STOP to determine if the target wants to disconnect the current transaction.

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

TRDY

Target Ready **Input/Output**

TRDY indicates the ability of the target of the transaction to complete the current data phase. Wait states are inserted until both IRDY and TRDY are asserted simultaneously. A data phase is completed on any clock when both IRDY and TRDY are asserted.

When the Am79C972 controller is a bus master, it checks TRDY during all read data phases to determine if valid data is present on AD[31:0]. During all write data phases, the device checks TRDY to determine if the target is ready to accept the data.

When the Am79C972 controller is the target of a transaction, it asserts TRDY during all read data phases to indicate that valid data is present on AD[31:0]. During all write data phases, the device asserts TRDY to indicate that it is ready to accept the data.

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

PME

Power Management Event **Output, Open Drain**

PME is an output that can be used to indicate that a power management event (a Magic Packet, an OnNow pattern match, or a change in link state) has been detected. The PME pin is asserted when either

1. PME_STATUS and PME_EN are both 1,
2. PME_EN_OVR and MPMAT are both 1, or
3. PME_EN_OVR and LCDET are both 1.

The PME signal is asynchronous with respect to the PCI clock.

Board Interface

Note: *Before programming the LED pins, see the description of LEDPE in BCR2, bit 12.*

LED0

LED0 **Output**

This output is designed to directly drive an LED. By default, LED0 indicates an active link connection. This pin can also be programmed to indicate other network status (see BCR4). The LED0 pin polarity is programmable, but by default it is active LOW. When the LED0 pin polarity is programmed to active LOW, the output is an open drain driver. When the LED0 pin polarity is programmed to active HIGH, the output is a totem pole driver.

Note: *The LED0 pin is multiplexed with the EEDI pin.*

LED1

LED1 **Output**

This output is designed to directly drive an LED. By default, LED1 indicates receive activity on the network. This pin can also be programmed to indicate other network status (see BCR5). The LED1 pin polarity is programmable, but by default, it is active LOW. When the LED1 pin polarity is programmed to active LOW, the output is an open drain driver. When the LED1 pin po-

larity is programmed to active HIGH, the output is a totem pole driver.

Note: *The $\overline{\text{LED1}}$ pin is multiplexed with the EESK and SFBD pins.*

The $\overline{\text{LED1}}$ pin is also used during EEPROM Auto-Detection to determine whether or not an EEPROM is present at the Am79C972 controller interface. At the last rising edge of CLK while $\overline{\text{RST}}$ is active LOW, $\overline{\text{LED1}}$ is sampled to determine the value of the EEDET bit in BCR19. It is important to maintain adequate hold time around the rising edge of the CLK at this time to ensure a correctly sampled value. A sampled HIGH value means that an EEPROM is present, and EEDET will be set to 1. A sampled LOW value means that an EEPROM is not present, and EEDET will be set to 0. 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.

WARNING: The input signal level of $\overline{\text{LED1}}$ must be insured for correct EEPROM detection before the deassertion of $\overline{\text{RST}}$.

$\overline{\text{LED2}}$

LED2 **Output**

This output is designed to directly drive an LED. This pin can be programmed to indicate various network status (see BCR6). The $\overline{\text{LED2}}$ pin polarity is programmable, but by default it is active LOW. When the $\overline{\text{LED2}}$ pin polarity is programmed to active LOW, the output is an open drain driver. When the $\overline{\text{LED2}}$ pin polarity is programmed to active HIGH, the output is a totem pole driver.

Note: *The $\overline{\text{LED2}}$ pin is multiplexed with the SRDCLK pin and the MIIRXFRTGE pins.*

$\overline{\text{LED3}}$

LED3 **Output**

This output is designed to directly drive an LED. By default, $\overline{\text{LED3}}$ indicates transmit activity on the network. This pin can also be programmed to indicate other network status (see BCR7). The $\overline{\text{LED3}}$ pin polarity is programmable, but by default it is active LOW. When the $\overline{\text{LED3}}$ pin polarity is programmed to active LOW, the output is an open drain driver. When the $\overline{\text{LED3}}$ pin polarity is programmed to active HIGH, the output is a totem pole driver.

Special attention must be given to the external circuitry attached to this pin. When this pin is used to drive an LED while an EEPROM is used in the system, then buffering maybe required between the $\overline{\text{LED3}}$ pin and the LED circuit. If an LED circuit were directly attached to this pin, it may create an I_{OL} requirement that could not be met by the serial EEPROM attached to this pin.

If no EEPROM is included in the system design or low current LEDs are used, then the $\overline{\text{LED3}}$ signal may be directly connected to an LED without buffering. For more details regarding LED connection, see the section on *LED Support*.

Note: *The $\overline{\text{LED3}}$ pin is multiplexed with the EEDO, SRD, MIIRXFRTGD pins.*

PG

Power Good **Input**

The PG pin has two functions: (1) it puts the device into Magic Packet™ mode, and (2) it blocks any resets when the PCI bus power is off.

When PG is LOW and either MPPEN or MPMODE is set to 1, the device enters the Magic Packet mode.

When PG is LOW, a LOW assertion of the PCI $\overline{\text{RST}}$ pin will only cause the PCI interface pins (except for $\overline{\text{PME}}$) to be put in the high impedance state. The internal logic will ignore the assertion of $\overline{\text{RST}}$.

When PG is HIGH, assertion of the PCI $\overline{\text{RST}}$ pin causes the controller logic to be reset and the configuration information to be loaded from the EEPROM.

PG input should be kept high during the NAND tree testing.

RWU

Remote Wake Up **Output**

RWU is an output that is asserted either when the controller is in the Magic Packet mode and a Magic Packet frame has been detected, or the controller is in the Link Change Detect mode and a Link Change has been detected.

This pin can drive the external system management logic that causes the CPU to get out of a low power mode of operation. This pin is implemented for designs that do not support the $\overline{\text{PME}}$ function.

Three bits that are loaded from the EEPROM into CSR116 can program the characteristics of this pin:

1. RWU_POL determines the polarity of the RWU signal.
2. If RWU_GATE bit is set, RWU is forced to the high impedance state when PG input is LOW.
3. RWU_DRIVER determines whether the output is open drain or totem pole.

The internal power-on-reset signal forces this output into the high impedance state until after the polarity and drive type have been determined.

$\overline{\text{WUMI}}$

Wake-Up Mode Indicator **Output**

This output, which is capable of driving an LED, is asserted when the device is in Magic Packet mode. It can

be used to drive external logic that switches the device power source from the main power supply to an auxiliary power supply.

TBC_EN

Time Base Clock Enable **Input**

TBC_EN is an input that controls the selection of the source of the Time Base Clock. The Time Base Clock is used in loading the EEPROM, generation of the PHY_RST, and the timing of the MDC and MDIO signals. When the input to this pin is LOW, an internal free running oscillator with a maximum frequency of 20 MHz is used. When the input to this pin is HIGH, the TBC_IN pin input is used to inject externally generated clock into the device. For typical applications which will use the internal oscillation, this pin should be tied to ground.

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

TBC_IN

Time Base Clock Input **Input**

TBC_IN may be used to connect to an external clock source to drive the internal circuitry that loads the EEPROM and controls the MDC and MDIO signals. This input is selected when the TBC_EN pin is HIGH. This pin should be tied to ground when the TBC_EN pin is LOW.

PHY_RST

PHY Reset **Output**

PHY_RST is an output pin that is used to reset the external PHY. This output eliminates the need for a fan out buffer for the PCI RST signal, provides polarity for the specific PHY used, and prevents the resetting of the PHY when the PG input is LOW. The output polarity is determined by the RST_POL bit(CSR116, bit0).

EEPROM Interface

EECS

EEPROM Chip Select **Output**

This pin is designed to directly interface to a serial EEPROM that uses the 93C46 EEPROM interface protocol. EECS is connected to the EEPROM's chip select pin. It is controlled by either the Am79C972 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**

This pin is designed to directly interface to a serial EEPROM that uses the 93C46 EEPROM interface protocol. EEDI is connected to the EEPROM's data input pin. It is controlled by either the Am79C972 controller during command portions of a read of the entire EEPROM, or indirectly by the host system by writing to BCR19, bit 0.

Note: *The EEDI pin is multiplexed with the $\overline{LED0}$ pin.*

EEDO

EEPROM Data Out **Input**

This pin is designed to directly interface to a serial EEPROM that uses the 93C46 EEPROM interface protocol. EEDO is connected to the EEPROM's data output pin. It is controlled by either the Am79C972 controller during command portions of a read of the entire EEPROM, or indirectly by the host system by reading from BCR19, bit 0.

Note: *The EEDO pin is multiplexed with the $\overline{LED3}$, MIIRXFRTGD, and SRD pins.*

EESK

EEPROM Serial Clock **Input/Output**

This pin is designed to directly interface to a serial EEPROM that uses the 93C46 EEPROM interface protocol. EESK is connected to the EEPROM's clock pin. It is controlled by either the Am79C972 controller directly during a read of the entire EEPROM, or indirectly by the host system by writing to BCR19, bit 1.

Note: *The EESK pin is multiplexed with the $\overline{LED1}$ and SFBD pins.*

The EESK pin is also used during EEPROM Auto-Detection to determine whether or not an EEPROM is present at the Am79C972 controller interface. At the rising edge of the last CLK edge while $\overline{RST}$ is asserted, EESK 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 1. A sampled LOW value means that an EEPROM is not present, and EEDET will be set to 0. 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 to resolve the EEDET setting.

WARNING: The input signal level of EESK must be valid for correct EEPROM detection before the deassertion of $\overline{RST}$.

Expansion Bus Interface

EBUA_EBA[7:0]

Expansion Bus Upper Address/
Expansion Bus Address [7:0] **Output**

The EBUA_EBA[7:0] pins provide the least and most significant bytes of address on the Expansion Bus. The most significant address byte (address bits [19:16] during boot device accesses) is valid on these pins at the beginning of a boot device access, at the rising edge of AS_$\overline{EBOE}$. This upper address byte must be stored ex-

ternally in a D flip-flop. During subsequent cycles of a boot device access, address bits [7:0] are present on these pins.

All EBUA_EBA[7:0] outputs are forced to a constant level to conserve power while no access on the Expansion Bus is being performed.

EBDA[15:8]

Expansion Bus Data/Address [15:8] Input/Output

When $\overline{\text{EROMCS}}$ is asserted low, EBDA[15:8] contain address bits [15:8] for boot device accesses.

The EBDA[15:8] signals are driven to a constant level to conserve power while no access on the Expansion Bus is being performed.

EBD[7:0]

Expansion Bus Data [7:0] Input/Output

The EBD[7:0] pins provide data bits [7:0] for EPROM/FLASH accesses. The EBD[7:0] signals are internally forced to a constant level to conserve power while no access on the Expansion Bus is being performed.

$\overline{\text{EROMCS}}$

Expansion ROM Chip Select Output

$\overline{\text{EROMCS}}$ serves as the chip select for the boot device. It is asserted low during the data phases of boot device accesses.

AS_$\overline{\text{EBOE}}$

Address Strobe/Expansion Bus Output Enable Output

AS_$\overline{\text{EBOE}}$ functions as the address strobe for the upper address bits on the EBUA_EBA[7:0] pins and as the output enable for the Expansion Bus.

As an address strobe, a rising edge on AS_$\overline{\text{EBOE}}$ is supplied at the beginning of boot device accesses. This rising edge provides a clock edge for a ‘374 D-type edge-triggered flip-flop which must store the upper address byte during Expansion Bus accesses for EPROM/Flash.

AS_$\overline{\text{EBOE}}$ is asserted active LOW during boot device read operations on the expansion bus and is deasserted during boot device write operations.

$\overline{\text{EBWE}}$

Expansion Bus Write Enable Output

$\overline{\text{EBWE}}$ provides the write enable for write accesses to the Flash device.

EBCLK

Expansion Bus Clock Input

EBCLK may be used as the fundamental clock to drive the Expansion Bus and internal SRAM access cycles. The actual internal clock used to drive the Expansion Bus cycles depends on the values of the EBCS and CLK_FAC settings in BCR27. Refer to the SRAM Interface Bandwidth Requirements section for details on determining the required EBCLK frequency. If a clock source other than the EBCLK pin is programmed (BCR27, bits 5:3) to be used to run the Expansion Bus interface, this input should be tied to VDD through a 4.7 kΩ resistor.

EBCLK is not used to drive the bus interface, internal buffer management unit, or the network functions.

Media Independent Interface

TX_CLK

Transmit Clock Input

TX_CLK is a continuous clock input that provides the timing reference for the transfer of the TX_EN, TXD[3:0], and TX_ER signals out of the Am79C972 device. TX_CLK must provide a nibble rate clock (25% of the network data rate). Hence, an MII transceiver operating at 10 Mbps must provide a TX_CLK frequency of 2.5 MHz and an MII transceiver operating at 100 Mbps must provide a TX_CLK frequency of 25 MHz.

Note: *The TX_CLK pin is multiplexed with the TXCLK pin.*

TXD[3:0]

Transmit Data Output

TXD[3:0] is the nibble-wide MII transmit data bus. Valid data is generated on TXD[3:0] on every TX_CLK rising edge while TX_EN is asserted. While TX_EN is deasserted, TXD[3:0] values are driven to a 0. TXD[3:0] transitions synchronous to TX_CLK rising edges.

Note: *The TXD[0] pin is multiplexed with the TXD pin.*

TX_EN

Transmit Enable Output

TX_EN indicates when the Am79C972 device is presenting valid transmit nibbles on the MII. While TX_EN is asserted, the Am79C972 device generates TXD[3:0] and TX_ER on TX_CLK rising edges. TX_EN is asserted with the first nibble of preamble and remains asserted throughout the duration of a packet until it is deasserted prior to the first TX_CLK following the final nibble of the frame. TX_EN transitions synchronous to TX_CLK rising edges.

Note: *The TX_EN pin is multiplexed with the TXEN pin.*

TX_ER

Transmit Error Output

TX_ER is an output that, if asserted while TX_EN is asserted, instructs the MII PHY device connected to the Am79C972 device to transmit a code group error. TX_ER is unused and is reserved for future use and will always be driven to a logical zero.

COL

Collision **Input**

COL is an input that indicates that a collision has been detected on the network medium.

Note: *The COL pin is multiplexed with the CLSN pin.*

CRS

Carrier Sense **Input**

CRS is an input that indicates that a non-idle medium, due either to transmit or receive activity, has been detected.

Note: *The CRS pin is multiplexed with the RXEN pin.*

RX_CLK

Receive Clock **Input**

RX_CLK is a clock input that provides the timing reference for the transfer of the RX_DV, RXD[3:0], and RX_ER signals into the Am79C972 device. RX_CLK must provide a nibble rate clock (25% of the network data rate). Hence, an MII transceiver operating at 10 Mbps must provide an RX_CLK frequency of 2.5 MHz and an MII transceiver operating at 100 Mbps must provide an RX_CLK frequency of 25 MHz. When the external PHY switches the RX_CLK and TX_CLK, it ***must*** provide glitch-free clock pulses.

Note: *The RX_CLK pin is multiplexed with the RXCLK pin.*

RXD[3:0]

Receive Data **Input**

RXD[3:0] is the nibble-wide MII receive data bus. Data on RXD[3:0] is sampled on every rising edge of RX_CLK while RX_DV is asserted. RXD[3:0] is ignored while RX_DV is de-asserted.

Note: *The RXD[0] pin is multiplexed with the RXFRTGD pin.*

If the MII port is not selected, the RXD[3:0] pin can be left floating.

RX_DV

Receive Data Valid **Input**

RX_DV is an input used to indicate that valid received data is being presented on the RXD[3:0] pins and RX_CLK is synchronous to the receive data. In order for a frame to be fully received by the Am79C972 device on the MII, RX_DV must be asserted prior to the RX_CLK rising edge, when the first nibble of the Start of Frame Delimiter is driven on RXD[3:0], and must remain asserted until after the rising edge of RX_CLK, when the last nibble of the CRC is driven on RXD[3:0]. RX_DV must then be deasserted prior to the RX_CLK rising edge which follows this final nibble. RX_DV transitions are synchronous to RX_CLK rising edges.

Note: *The RX_DV pin is multiplexed with the RXFRTGE pin.*

If the MII port is not selected, the RX_DV pin can be left floating.

RX_ER

Receive Error **Input**

RX_ER is an input that indicates that the MII transceiver device has detected a coding error in the receive frame currently being transferred on the RXD[3:0] pins. When RX_ER is asserted while RX_DV is asserted, a CRC error will be indicated in the receive descriptor for the incoming receive frame. RX_ER is ignored while RX_DV is deasserted. Special code groups generated on RXD while RX_DV is deasserted are ignored (e.g., Bad SSD in TX and IDLE in T4). RX_ER transitions are synchronous to RX_CLK rising edges.

Note: *The RX_ER pin is multiplexed with the RXDAT pin.*

MDC

Management Data Clock **Output**

MDC is a non-continuous clock output that provides a timing reference for bits on the MDIO pin. During MII management port operations, MDC runs at a nominal frequency of 2.5 MHz. When no management operations are in progress, MDC is driven LOW. The MDC is derived from the Time Base Clock.

If the MII port is not selected, the MDC pin can be left floating.

MDIO

Management Data I/O **Input/Output**

MDIO is the bidirectional MII management port data pin. MDIO is an output during the header portion of the management frame transfers and during the data portions of write transfers. MDIO is an input during the data portions of read data transfers. When an operation is not in progress on the management port, MDIO is not driven. MDIO transitions from the Am79C972 controller are synchronous to MDC falling edges.

If the PHY is attached through an MII physical connector, then the MDIO pin should be externally pulled down to V_{SS} with a 10-kΩ $\pm$5% resistor. If the PHY is on board, then the MDIO pin should be externally pulled up to V_{CC} with a 10-kΩ $\pm$5% resistor.

General Purpose Serial Interface

CLSN

Collision **Input**

CLSN is an input that indicates a collision has occurred on the network.

Note: *The CLSN pin is multiplexed with the COL pin.*

RXCLK

Receive Clock **Input**

RXCLK is an input. The rising edges of the RXCLK signal are used to sample the data on the RXDAT input whenever the RXEN input is HIGH.

Note: *The RXCLK pin is multiplexed with the RX_CLK pin.*

RXDAT

Receive Data **Input**

RXDAT is an input. The rising edges of the RXCLK signal are used to sample the data on the RXDAT input whenever the RXEN input is HIGH.

Note: *The RXDAT pin is multiplexed with the RX_ER pin.*

RXEN

Receive Enable **Input**

RXEN is an input. When this signal is HIGH, it indicates to the core logic that the data on the RXDAT input pin is valid.

Note: *The RXEN pin is multiplexed with the CRS pin.*

TXCLK

Transmit Clock **Input**

TXCLK is an input that provides a clock signal for MAC activity, both transmit and receive. The rising edges of the TXCLK can be used to validate TXDAT output data.

Note: *The TXCLK pin is multiplexed with the TX_CLK pin.*

TXDAT

Transmit Data **Output**

TXDAT is an output that provides the serial bit stream for transmission, including preamble, SFD, data, and FCS field, if applicable.

Note: *The TXDAT pin is multiplexed with the TXD[0] pin.*

TXEN

Transmit Enable **Output**

TXEN is an output that provides an enable signal for transmission. Data on the TXDAT pin is not valid unless the TXEN signal is HIGH.

Note: *The TXEN pin is multiplexed with the TX_EN pin.*

External Address Detection Interface

$\overline{\text{EAR}}$

External Address Reject Low **Input**

The incoming frame will be checked against the internally active address detection mechanisms and the result of this check will be OR'd with the value on the $\overline{\text{EAR}}$ pin. The $\overline{\text{EAR}}$ pin is defined as $\overline{\text{REJECT}}$. The pin value is OR'd with the internal address detection result to determine if the current frame should be accepted or rejected.

The $\overline{\text{EAR}}$ pin ***must not*** be left unconnected, it should be tied to VDD through a 10-kΩ $\pm$5% resistor.

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

SFBD

Start Frame-Byte Delimiter **Output**

For the GPSI port during External Address Detection:

An initial rising edge on the SFBD signal indicates that a start of frame delimiter has been detected. The serial bit stream will follow on the SRD signal, commencing with the destination address field. SFBD will go high for 4 bit times (400 ns when operating at 10 Mbps) after detecting the second "1" in the SFD (Start of Frame Delimiter) of a received frame. SFBD will subsequently toggle every 4 bit times (1.25 MHz frequency when operating at 10 Mbps) with each rising edge indicating the first bit of each subsequent byte of the received serial bit stream.

For the External PHY attached to the Media Independent Interface during External Address Detection:

An initial rising edge on the SFBD signal indicates that a start of valid data is present on the RXD[3:0] pins. SFBD will go high for one nibble time (400 ns when operating at 10 Mbps and 40 ns when operating at 100 Mbps) one RX_CLK period after RX_DV has been asserted and RX_ER is deasserted and the detection of the SFD (Start of Frame Delimiter) of a received frame. Data on the RXD[3:0] will be the start of the destination address field. SFBD will subsequently toggle every nibble time (1.25 MHz frequency when operating at 10 Mbps and 12.5 MHz frequency when operating at 100 Mbps) indicating the first nibble of each subsequent byte of the received nibble stream. The RX_CLK should be used in conjunction with the SFBD to latch the correct data for external address matching. SFBD will be active only during frame reception.

Note: *The SFBD pin is multiplexed with the EESK and $\overline{LED1}$ pins.*

SRD

Serial Receive Data **Input/Output**

SRD is the decoded NRZ data from the network when in GPSI mode. This signal can be used for external address detection.

Note: *When the MII port is selected, SRD will not generate transitions and receive data must be derived from the Media Independent Interface RXD[3:0] pins.*

Note also that the SRD pin is multiplexed with the MIIRXFRTGD, EEDO, and $\overline{LED3}$ pins.

SRDCLK

Serial Receive Data Clock **Output**

Serial Receive Data is synchronous with reference to SRDCLK.

Note: *When the MII port is selected, SRDCLK will not generate transitions and the receive clock must be derived from the MII RX_CLK pin.*

Note also that the SRDCLK pin is multiplexed with the MIIRXFRTGE and $\overline{LED2}$ pins.

RXFRTGD

Receive Frame Tag Data **Input**

When the EADI is enabled (EADISEL, BCR2, bit 3), the Receive Frame Tagging is enabled (RXFRTG, CSR7, bit 14), and the MII is not selected, the RXFRTGD pin becomes a data input pin for the Receive Frame Tag. See the *Receive Frame Tagging* section for details.

Note: *The RXFRTGD pin is multiplexed with the RXD[0] pin.*

RXFRTGE

Receive Frame Tag Enable **Input**

When the EADI is enabled (EADISEL, BCR2, bit 3), the Receive Frame Tagging is enabled (RXFRTG, CSR7, bit 14), and the MII is not selected, the RXFRTGE pin becomes a data input enable pin for the Receive Frame Tag. See the *Receive Frame Tagging* section for details.

Note: *The RXFRTGE pin is multiplexed with the RX_DV pin.*

MIIRXFRTGD

MII Receive Frame Tag Enable **Input**

When the EADI is enabled (EADISEL, BCR2, bit 3), the Receive Frame Tagging is enabled (RXFRTG, CSR7, bit 14), and the MII is selected, the MIIRXFRTGD pin becomes a data input pin for the Receive Frame Tag. See the *Receive Frame Tagging* section for details.

Note: *The MIIRXFRTGD pin is multiplexed with the SRD, EEDO, and $\overline{LED3}$ pins.*

MIIRXFRTGE

MII Receive Frame Tag Enable **Input**

When the EADI is enabled (EADISEL, BCR2, bit 3), the Receive Frame Tagging is enabled (RXFRTG, CSR7, bit 14), and the MII is selected, the MIIRXFRTGE pin becomes a data input enable pin for the Receive Frame Tag. See the *Receive Frame Tagging* section for details.

Note: *The MIIRXFRTGE pin is multiplexed with the SRDCLK and $\overline{LED2}$ pins.*

IEEE 1149.1 (1990) Test Access Port Interface

TCK

Test Clock **Input**

TCK is the clock input for the boundary scan test mode operation. It can operate at a frequency of up to 10 MHz. TCK has an internal pull up resistor.

TDI

Test Data In **Input**

TDI is the test data input path to the Am79C972 controller. The pin has an internal pull up resistor.

TDO

Test Data Out **Output**

TDO is the test data output path from the Am79C972 controller. The pin is tri-stated when the JTAG port is inactive.

TMS

Test Mode Select **Input**

A serial input bit stream on the TMS pin is used to define the specific boundary scan test to be executed. The pin has an internal pull up resistor.

Power Supply Pins

VDDB

I/O Buffer Power (7 Pins) **Power**

There are seven power supply pins that are used by the input/output buffer drivers. All VDDB pins must be connected to a +3.3 V supply.

VDD_PCI

PCI I/O Buffer Power (9 Pins) **Power**

There are nine power supply pins that are used by the PCI input/output buffer drivers (except $\overline{\text{PME}}$ driver). All VDD_PCI pins must be connected to a +3.3 V supply.

VSSB

I/O Buffer Ground (17 Pins) **Power**

There are 17 ground pins that are used by the input/output buffer drivers.

VDD

Digital Power (6 Pins) **Power**

There are six power supply pins that are used by the internal digital circuitry. All VDD pins must be connected to a +3.3 V supply.

VSS

Digital Ground (8 Pins) **Power**

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

BASIC FUNCTIONS

System Bus Interface

The Am79C972 controller is designed to operate as a bus master during normal operations. Some slave I/O accesses to the Am79C972 controller are required in normal operations as well. Initialization of the Am79C972 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 Am79C972 controller. The EEPROM read operation is performed through the 93C46 EEPROM interface. The ISO 8802-3 (IEEE/ANSI 802.3) Ethernet Address may reside within the serial EEPROM. Some Am79C972 controller configuration registers may also be programmed by the EEPROM read operation.

The Address PROM, on-chip board-configuration registers, and the Ethernet controller registers occupy 32 bytes of address space. I/O and memory mapped I/O accesses are supported. Base Address registers in the PCI configuration space allow locating the address space on a wide variety of starting addresses.

For diskless stations, the Am79C972 controller supports a ROM or Flash-based (both referred to as the *Expansion ROM* throughout this specification) boot device of up to 1 Mbyte in size. The host can map the boot device to any memory address that aligns to a 1-Mbyte boundary by modifying the Expansion ROM Base Address register in the PCI configuration space.

Software Interface

The software interface to the Am79C972 controller is divided into three parts. One part is the PCI configuration registers used to identify the Am79C972 controller and to setup the configuration of the device. The setup information includes the I/O or memory mapped I/O base address, mapping of the Expansion ROM, and the routing of the Am79C972 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 Am79C972 controller. The Am79C972 controller occupies 32 bytes of address space that must begin on a 32-byte block boundary. The address space can be mapped into I/O or memory space (memory mapped I/O). The I/O Base Address Register in the PCI Configuration Space controls the start address of the address space if it is mapped to I/O space. The Memory Mapped I/O Base Address Register controls the start address of the address space if it is mapped to memory space. The 32-byte address space is used by the software to program the Am79C972 controller operating mode, to enable and disable various features, to monitor operating status, and to request particular functions to be executed by the Am79C972 controller.

The third portion of the software interface is the descriptor and buffer areas that are shared between the software and the Am79C972 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 frame data, and it is used to transfer frame status from the Am79C972 controller to the software. The buffer areas are locations that hold frame data for transmission or that accept frame data that has been received.

Network Interfaces

The Am79C972 controller can be connected to an IEEE 802.3 or proprietary network via one of two network interfaces. The Media Independent Interface (MII) provides an IEEE 802.3-compliant nibble-wide interface to an external 100- and/or 10-Mbps transceiver device. The General Purpose Serial Interface (GPSI) is functionally equivalent to the GPSI found on the LANCE.

While in auto-selection mode, the interface in use is determined by the Network Port Manager. If the quiescent state of the MII MDIO pin is HIGH, the MII is activated. The GPSI port can only be enabled by disabling the auto-selection and manually selecting the GPSI as the network port.

The Am79C972 controller supports both half-duplex and full-duplex operation on network interfaces (i.e., GPSI and MII).

DETAILED FUNCTIONS

Slave Bus Interface Unit

The slave bus interface unit (BIU) controls all accesses to the PCI configuration space, the Control and Status Registers (CSR), the Bus Configuration Registers (BCR), the Address PROM (APROM) locations, and the Expansion ROM. Table 2 shows the response of the Am79C972 controller to each of the PCI commands in slave mode.

Table 2. Slave Commands

C[3:0]	Command	Use
0000	Interrupt Acknowledge	Not used
0001	Special Cycle	Not used
0010	I/O Read	Read of CSR, BCR, APROM, and Reset registers
0011	I/O Write	Write to CSR, BCR, and APROM
0100	Reserved	
0101	Reserved	
0110	Memory Read	Memory mapped I/O read of CSR, BCR, APROM, and Reset registers Read of the Expansion Bus
0111	Memory Write	Memory mapped I/O write of CSR, BCR, and APROM
1000	Reserved	
1001	Reserved	
1010	Configuration Read	Read of the Configuration Space
1011	Configuration Write	Write to the Configuration Space
1100	Memory Read Multiple	Aliased to Memory Read
1101	Dual Address Cycle	Not used
1110	Memory Read Line	Aliased to Memory Read
1111	Memory Write Invalidate	Aliased to Memory Write

Slave Configuration Transfers

The host can access the Am79C972 PCI configuration space with a configuration read or write command. The Am79C972 controller will assert DEVSEL during the address phase when IDSEL is asserted, AD[1:0] are both 0, and the access is a configuration cycle. AD[7:2] select the DWord location in the configuration space. The Am79C972 controller ignores AD[10:8], because it is a single function device. AD[31:11] are don't care.

AD31 AD11	AD10 AD8	AD7 AD2	AD1	AD0
Don't care	Don't care	DWord index	0	0

The active bytes within a DWord are determined by the byte enable signals. Eight-bit, 16-bit, and 32-bit transfers are supported. DEVSEL is asserted two clock cycles after the host has asserted FRAME. All configuration cycles are of fixed length. The Am79C972 controller will assert TRDY on the third clock of the data phase.

The Am79C972 controller does not support burst transfers for access to configuration space. When the host keeps FRAME asserted for a second data phase, the Am79C972 controller will disconnect the transfer.

When the host tries to access the PCI configuration space while the automatic read of the EEPROM after H_RESET (see section on RESET) is on-going, the Am79C972 controller will terminate the access on the PCI bus with a disconnect/retry response.

The Am79C972 controller supports fast back-to-back transactions to different targets. This is indicated by the Fast Back-To-Back Capable bit (PCI Status register, bit 7), which is hardwired to 1. The Am79C972 controller is capable of detecting a configuration cycle even when its address phase immediately follows the data phase of a transaction to a different target without any idle state in-between. There will be no contention on the DEVSEL, TRDY, and STOP signals, since the Am79C972 controller asserts DEVSEL on the second clock after FRAME is asserted (medium timing).

Slave I/O Transfers

After the Am79C972 controller is configured as an I/O device by setting IOEN (for regular I/O mode) or MEMEN (for memory mapped I/O mode) in the PCI Command register, it starts monitoring the PCI bus for access to its CSR, BCR, or APROM locations. If configured for regular I/O mode, the Am79C972 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 Am79C972 controller asserts DEVSEL if it detects an address match and the access is an I/O cycle. If configured for memory mapped I/O mode, the Am79C972 controller will look for an address that falls within its 32 bytes of memory address space (starting from the memory mapped I/O base address). The Am79C972 controller asserts DEVSEL if it detects an address match and the access is a memory cycle. DEVSEL is asserted two clock cycles after the host has asserted FRAME. See Figure 1 and Figure 2.

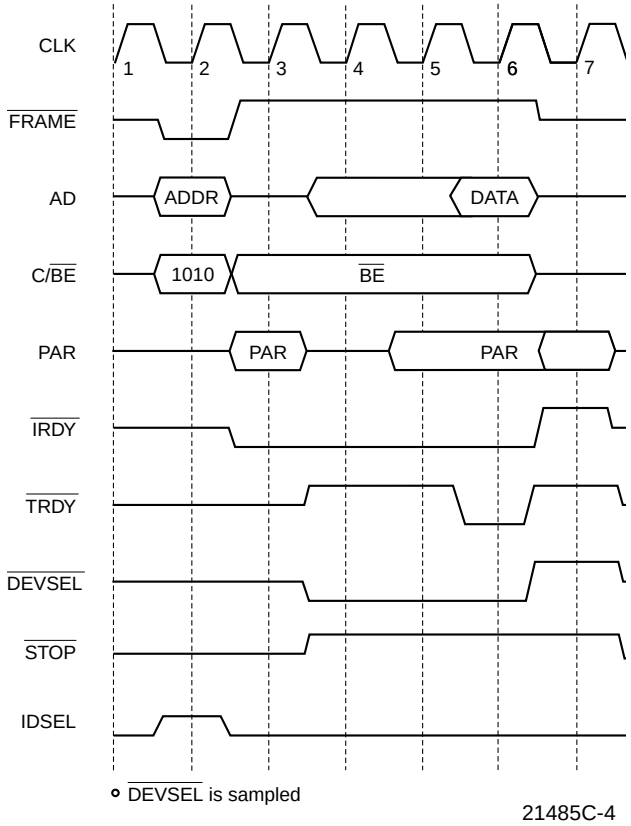


Figure 1. Slave Configuration Read

The Am79C972 controller will not assert DEVSEL if it detects an address match, but the PCI command is not of the correct type. In memory mapped I/O mode, the Am79C972 controller aliases all accesses to the I/O resources of the command types *Memory Read Multiple* and *Memory Read Line* to the basic Memory Read command. All accesses of the type *Memory Write and Invalidate* are aliased to the basic Memory Write command. Eight-bit, 16-bit, and 32-bit non-burst transactions are supported. The Am79C972 controller decodes all 32 address lines to determine which I/O resource is accessed.

The typical number of wait states added to a slave I/O or memory mapped I/O read or write access on the part of the Am79C972 controller is six to seven 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 Am79C972 controller does not support burst transfers for access to its I/O resources. When the host keeps FRAME asserted for a second data phase, the Am79C972 controller will disconnect the transfer.

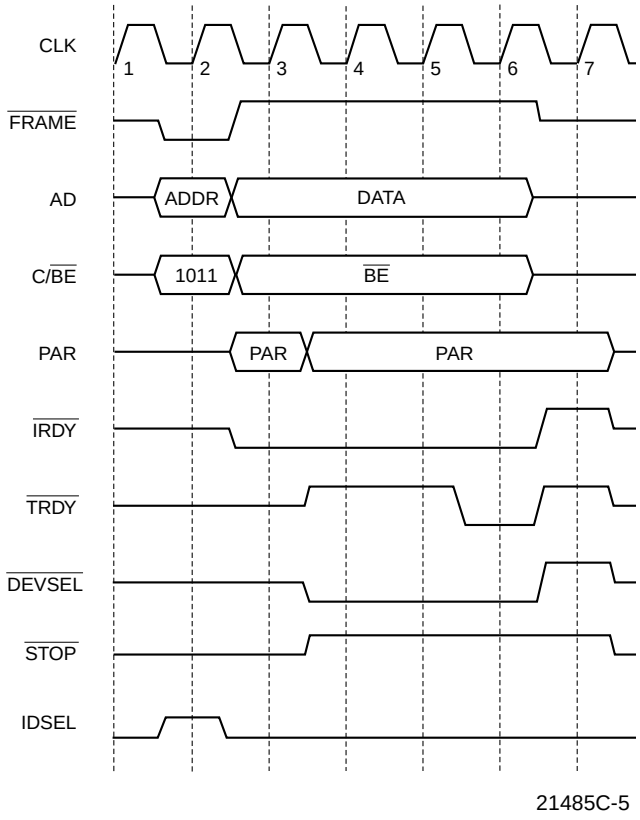
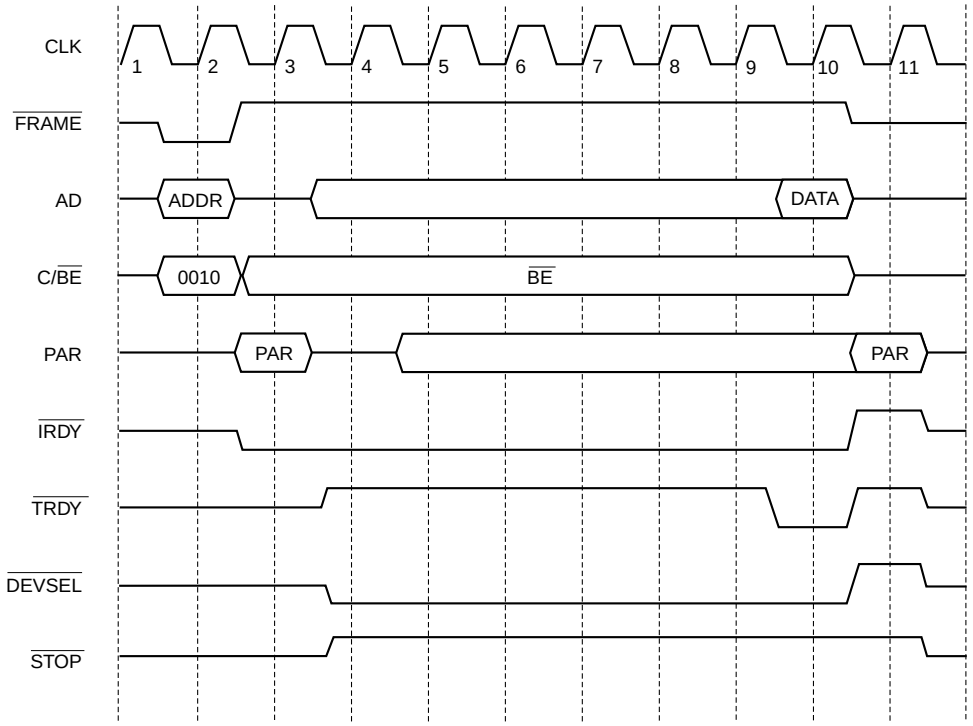


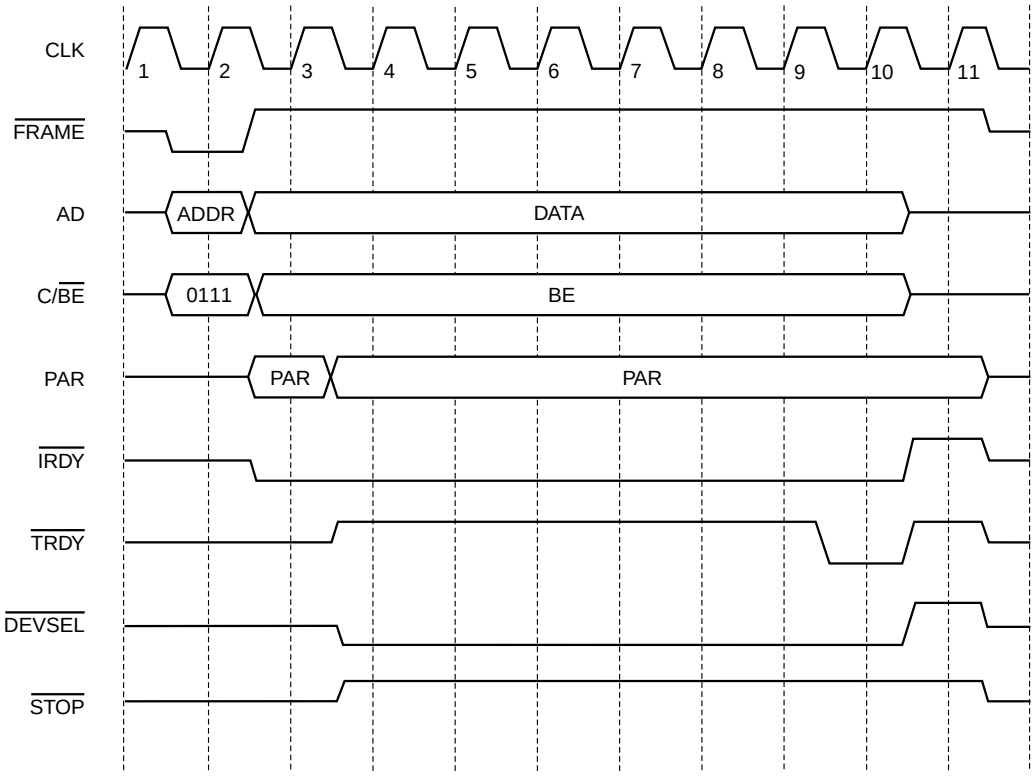
Figure 2. Slave Configuration Write

The Am79C972 controller supports fast back-to-back transactions to different targets. This is indicated by the Fast Back-To-Back Capable bit (PCI Status register, bit 7), which is hardwired to 1. The Am79C972 controller is capable of detecting an I/O or a memory-mapped I/O cycle even when its address phase immediately follows the data phase of a transaction to a different target, without any idle state in-between. There will be no contention on the DEVSEL, TRDY, and STOP signals, since the Am79C972 controller asserts DEVSEL on the second clock after FRAME is asserted (medium timing) See Figure 3 and Figure 4.



21485C-6

Figure 3. Slave Read Using I/O Command



21485C-7

Figure 4. Slave Write Using Memory Command

Expansion ROM Transfers

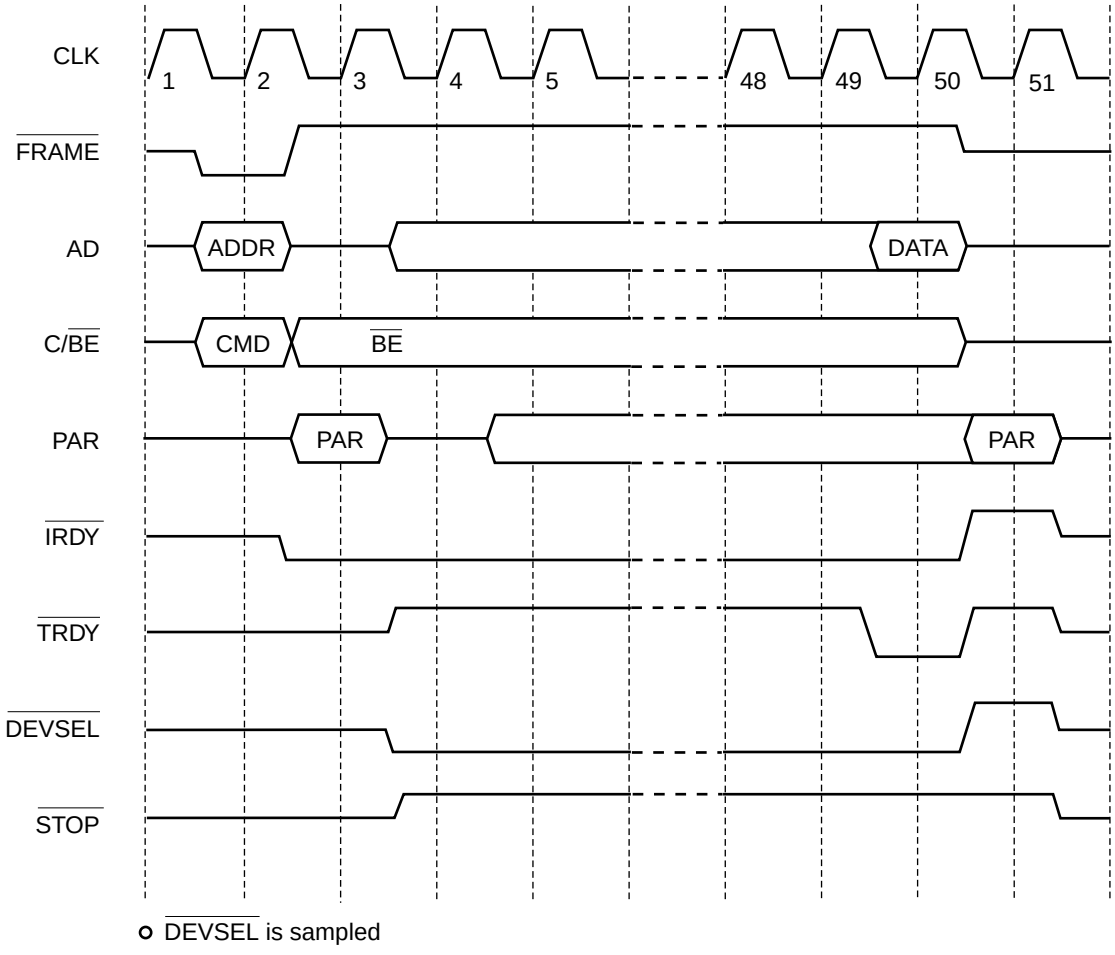
The host must initialize the Expansion ROM Base Address register at offset 30H in the PCI configuration space with a valid address before enabling the access to the device. The Am79C972 controller will not react to any access to the Expansion ROM until both MEMEN (PCI Command register, bit 1) and ROMEN (PCI Expansion ROM Base Address register, bit 0) are set to 1. After the Expansion ROM is enabled, the Am79C972 controller will assert DEVSEL on all memory read accesses with an address between ROMBASE and ROMBASE + 1M - 4. The Am79C972 controller aliases all accesses to the Expansion ROM of the command types *Memory Read Multiple* and *Memory Read Line* to the basic Memory Read command. Eight-bit, 16-bit, and 32-bit read transfers are supported.

Since setting MEMEN also enables memory mapped access to the I/O resources, attention must be given the PCI Memory Mapped I/O Base Address register before enabling access to the Expansion ROM. The host must set the PCI Memory Mapped I/O Base Address register to a value that prevents the Am79C972 controller from claiming any memory cycles not intended for it.

The Am79C972 controller will always read four bytes for every host Expansion ROM read access. TRDY will not be asserted until all four bytes are loaded into an internal scratch register. The cycle TRDY is asserted depends on the programming of the Expansion ROM interface timing. The following figure (Figure 5) assumes that ROMTMG (BCR18, bits 15-12) is at its default value.

Note: *The Expansion ROM should be read only during PCI configuration time for the PCI system.*

When the host tries to write to the Expansion ROM, the Am79C972 controller will claim the cycle by asserting DEVSEL. TRDY will be asserted one clock cycle later. The write operation will have no effect. Writes to the Expansion ROM are done through the BCR30 Expansion Bus Data Port. See the section on the *Expansion Bus Interface* for more details. See Figure 5.



21485C-8

Figure 5. Expansion ROM Read

During the boot procedure, the system will try to find an Expansion ROM. A PCI system assumes that an Expansion ROM is present when it reads the ROM signature 55H (byte 0) and AAH (byte 1).

Slave Cycle Termination

There are three scenarios besides normal completion of a transaction where the Am79C972 controller is the target of a slave cycle and it will terminate the access.

Disconnect When Busy

The Am79C972 controller cannot service any slave access while it is reading the contents of the EEPROM. Simultaneous access is not allowed in order to avoid conflicts, since the EEPROM is used to initialize some of the PCI configuration space locations and most of the BCRs and CSR116. The EEPROM read operation will always happen automatically after the deassertion of the $\overline{RST}$ pin. In addition, the host can start the read operation by setting the PREAD bit (BCR19, bit 14). While the EEPROM read is on-going, the Am79C972 controller will disconnect any slave access where it is the target by asserting $\overline{STOP}$ together with $\overline{DEVSEL}$, while driving $\overline{TRDY}$ high. $\overline{STOP}$ will stay asserted until the end of the cycle.

Note that I/O and memory slave accesses will only be disconnected if they are enabled by setting the IOEN or MEMEN bit in the PCI Command register. Without the enable bit set, the cycles will not be claimed at all. Since H_RESET clears the IOEN and MEMEN bits for the automatic EEPROM read after H_RESET, the disconnect only applies to configuration cycles.

A second situation where the Am79C972 controller will generate a PCI disconnect/retry cycle is when the host tries to access any of the I/O resources right after having read the Reset register. Since the access generates an internal reset pulse of about 1 µs in length, all further slave accesses will be deferred until the internal reset operation is completed. See Figure 6.

Disconnect Of Burst Transfer

The Am79C972 controller does not support burst access to the configuration space, the I/O resources, or to the Expansion Bus. The host indicates a burst transaction by keeping $\overline{FRAME}$ asserted during the data phase. When the Am79C972 controller sees $\overline{FRAME}$ and $\overline{IRDY}$ asserted in the clock cycle before it wants to assert $\overline{TRDY}$, it also asserts $\overline{STOP}$ at the same time. The transfer of the first data phase is still successful, since $\overline{IRDY}$ and $\overline{TRDY}$ are both asserted. See Figure 7.

.

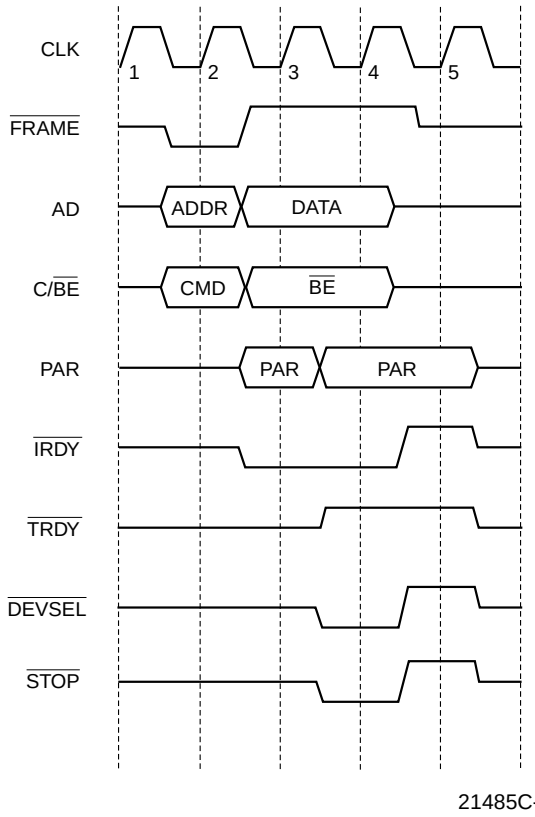


Figure 6. Disconnect Of Slave Cycle When Busy

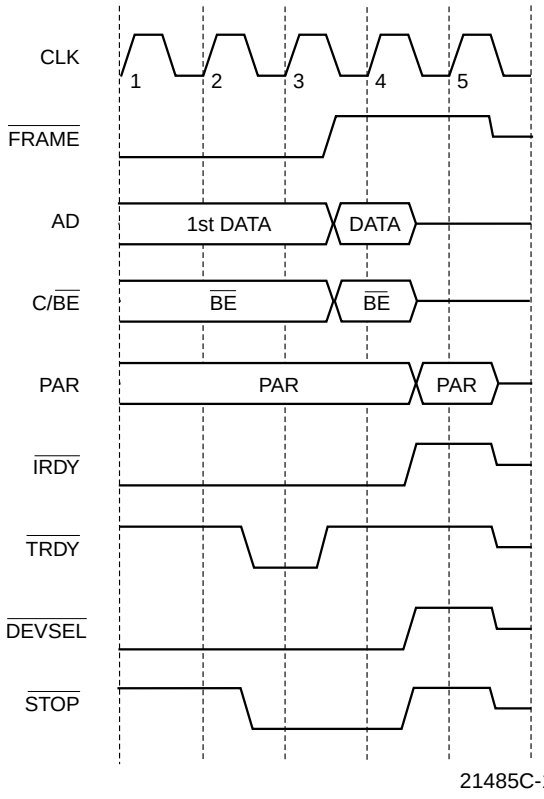


Figure 7. Disconnect Of Slave Burst Transfer - No Host Wait States

If the host is not yet ready when the Am79C972 controller asserts TRDY, the device will wait for the host to assert IRDY. When the host asserts IRDY and FRAME is still asserted, the Am79C972 controller will finish the first data phase by deasserting TRDY one clock later. At the same time, it will assert STOP to signal a disconnect to the host. STOP will stay asserted until the host removes FRAME. See Figure 8.

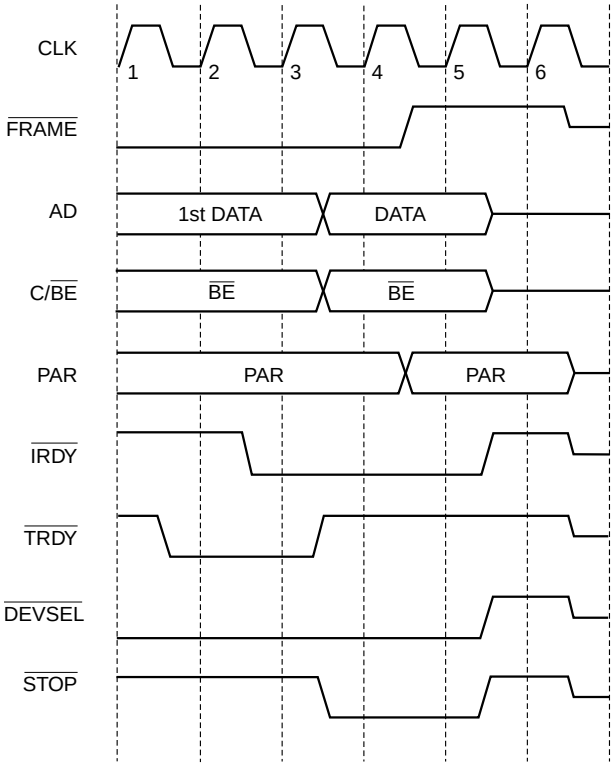


Figure 8. Disconnect Of Slave Burst Transfer - Host Inserts Wait States

Parity Error Response

When the Am79C972 controller is not the current bus master, it samples the AD[31:0], C/BE[3:0], and the PAR lines during the address phase of any PCI command for a parity error. When it detects an address parity error, the controller sets PERR (PCI Status register, bit 15) to 1. When reporting of that error is enabled by setting SERREN (PCI Command register, bit 8) and PERREN (PCI Command register, bit 6) to 1, the Am79C972 controller also drives the SERR signal low for one clock cycle and sets SERR (PCI Status register, bit 14) to 1. The assertion of SERR follows the address phase by two clock cycles. The Am79C972 controller will not assert DEVSEL for a PCI transaction that has an address parity error when PERREN and SERREN are set to 1. See Figure 9.

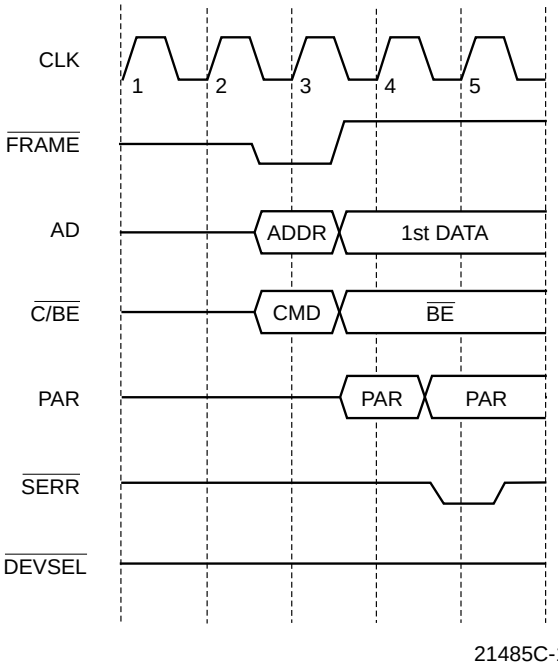


Figure 9. Address Parity Error Response

During the data phase of an I/O write, memory-mapped I/O write, or configuration write command that selects the Am79C972 controller as target, the device samples the AD[31:0] and C/BE[3:0] lines for parity on the clock edge, and data is transferred as indicated by the assertion of IRDY and TRDY. PAR is sampled in the following clock cycle. If a parity error is detected and reporting of that error is enabled by setting PERREN (PCI Command register, bit 6) to 1, PERR is asserted one clock later. The parity error will always set PERR (PCI Status register, bit 15) to 1 even when PERREN is cleared to 0. The Am79C972 controller will finish a transaction that has a data parity error in the normal way by asserting TRDY. The corrupted data will be written to the addressed location.

Figure 10 shows a transaction that suffered a parity error at the time data was transferred (clock 7, IRDY and TRDY are both asserted). PERR is driven high at the beginning of the data phase and then drops low due to the parity error on clock 9, two clock cycles after the data was transferred. After PERR is driven low, the Am79C972 controller drives PERR high for one clock cycle, since PERR is a sustained tri-state signal.

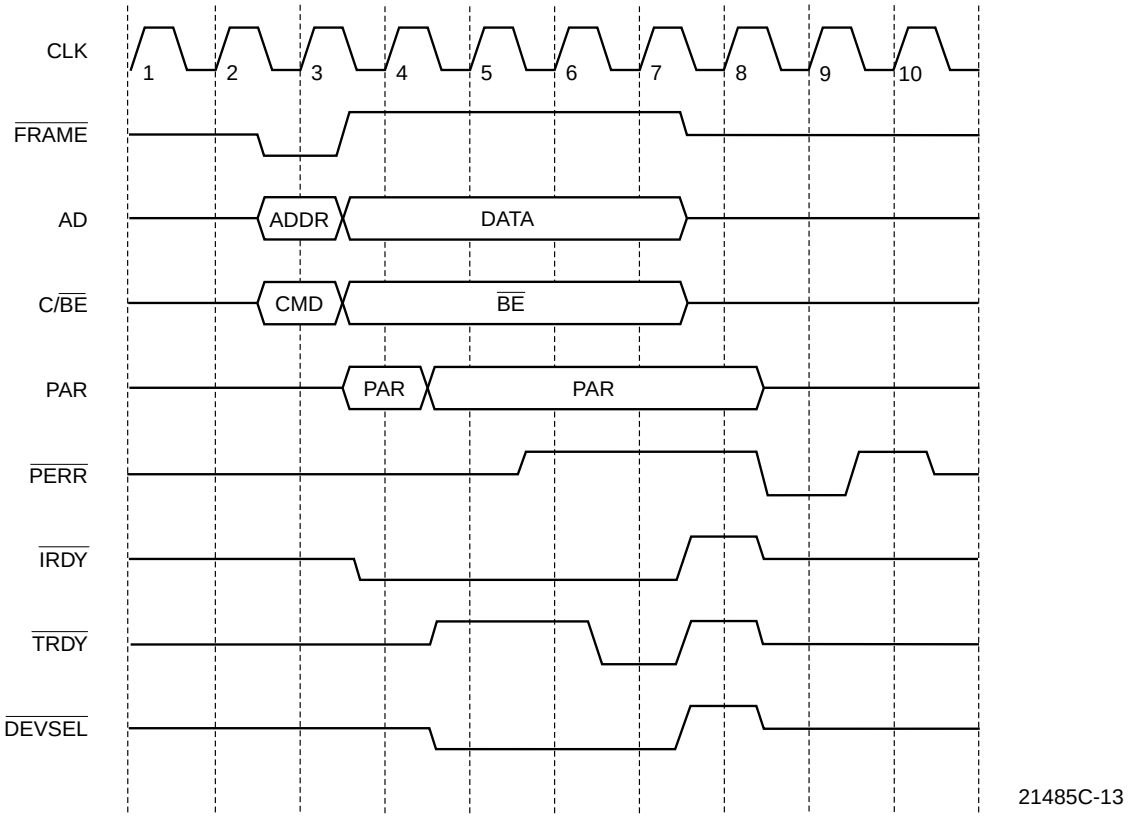


Figure 10. Slave Cycle Data Parity Error Response

Master Bus Interface Unit

The master Bus Interface Unit (BIU) controls the acquisition of the PCI bus and all accesses to the initialization block, descriptor rings, and the receive and transmit buffer memory. Table 3 shows the usage of PCI commands by the Am79C972 controller in master mode.

Table 3. Master Commands

C[3:0]	Command	Use
0000	Interrupt Acknowledge	Not used
0001	Special Cycle	Not used
0010	I/O Read	Not used
0011	I/O Write	Not used
0100	Reserved	
0101	Reserved	
0110	Memory Read	Read of the initialization block and descriptor rings Read of the transmit buffer in non-burst mode
0111	Memory Write	Write to the descriptor rings and to the receive buffer
1000	Reserved	
C[3:0]	**Command**	**Use**
1001	Reserved	
1010	Configuration Read	Not used
1011	Configuration Write	Not used
1100	Memory Read Multiple	Read of the transmit buffer in burst mode
1101	Dual Address Cycle	Not used
1110	Memory Read Line	Read of the transmit buffer in burst mode
1111	Memory Write Invalidate	Not used

Bus Acquisition

The Am79C972 microcode will determine when a DMA transfer should be initiated. The first step in any Am79C972 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 11 shows the Am79C972 controller bus acquisition. REQ is asserted and the arbiter returns GNT while another bus master is transferring data. The Am79C972 controller waits until the bus is idle (FRAME and IRDY deasserted) before it starts driving AD[31:0] and C/BE[3:0] on clock 5. FRAME is asserted at clock 5 indicating a valid address and command on AD[31:0] and C/BE[3:0]. The Am79C972 controller does not use address stepping which is reflected by ADSTEP (bit 7) in the PCI Command register being hardwired to 0.

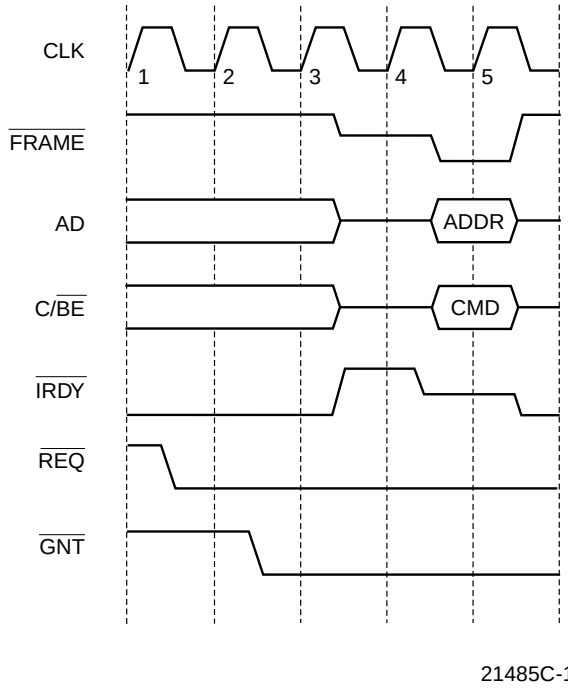


Figure 11. Bus Acquisition

In burst mode, the deassertion of REQ depends on the setting of EXTREQ (BCR18, bit 8). If EXTREQ is cleared to 0, REQ is deasserted at the same time as FRAME is asserted. (The Am79C972 controller never performs more than one burst transaction within a single bus mastership period.) If EXTREQ is set to 1, the Am79C972 controller does not deassert REQ until it starts the last data phase of the transaction.

Once asserted, REQ remains active until GNT has become active and independent of subsequent setting of STOP (CSR0, bit 2) or SPND (CSR5, bit 0). The assertion of H_RESET or S_RESET, however, will cause REQ to go inactive immediately.

Bus Master DMA Transfers

There are four primary types of DMA transfers. The Am79C972 controller uses non-burst as well as burst cycles for read and write access to the main memory.

Basic Non-Burst Read Transfer

By default, the Am79C972 controller uses non-burst cycles in all bus master read operations. All Am79C972 controller non-burst read accesses are of the PCI command type Memory Read (type 6). Note that during a non-burst read operation, all byte lanes will always be active. The Am79C972 controller will internally discard unneeded bytes.

The Am79C972 controller typically performs more than one non-burst read transaction within a single bus mastership period. FRAME is dropped between consecutive non-burst read cycles. REQ however stays asserted until FRAME is asserted for the last transaction. The Am79C972 controller supports zero wait state read cycles. It asserts IRDY immediately after the address phase and at the same time starts sampling DEVSEL. Figure 12 shows two non-burst read transactions. The first transaction has zero wait states. In the second transaction, the target extends the cycle by asserting TRDY one clock later.

Basic Burst Read Transfer

The Am79C972 controller supports burst mode for all bus master read operations. The burst mode must be enabled by setting BREADE (BCR18, bit 6). To allow burst transfers in descriptor read operations, the Am79C972 controller must also be programmed to use SWSTYLE 3 (BCR20, bits 7-0). All burst read accesses to the initialization block and descriptor ring are of the PCI command type Memory Read (type 6). Burst read accesses to the transmit buffer typically are longer than two data phases. When MEMCMD (BCR18, bit 9) is cleared to 0, all burst read accesses to the transmit buffer are of the PCI command type Memory Read Line (type 14). When MEMCMD (BCR18, bit 9) is set to1, all burst read accesses to the transmit buffer are of the PCI command type Memory Read Multiple (type 12). AD[1:0] will both be 0 during the address phase indicating a linear burst order. Note that during a burst read operation, all byte lanes will always be active. The Am79C972 controller will internally discard unneeded bytes.

The Am79C972 controller will always perform only a single burst read transaction per bus mastership period, where *transaction* is defined as one address phase and one or multiple data phases. The Am79C972 controller supports zero wait state read cycles. It asserts IRDY immediately after the address phase and at the same time starts sampling DEVSEL. FRAME is deasserted when the next to last data phase is completed.

Figure 13 shows a typical burst read access. The Am79C972 controller arbitrates for the bus, is granted access, reads three 32-bit words (DWord) from the system memory, and then releases the bus. In the example, the memory system extends the data phase of each access by one wait state. The example assumes that EXTREQ (BCR18, bit 8) is cleared to 0, therefore, REQ is deasserted in the same cycle as FRAME is asserted.

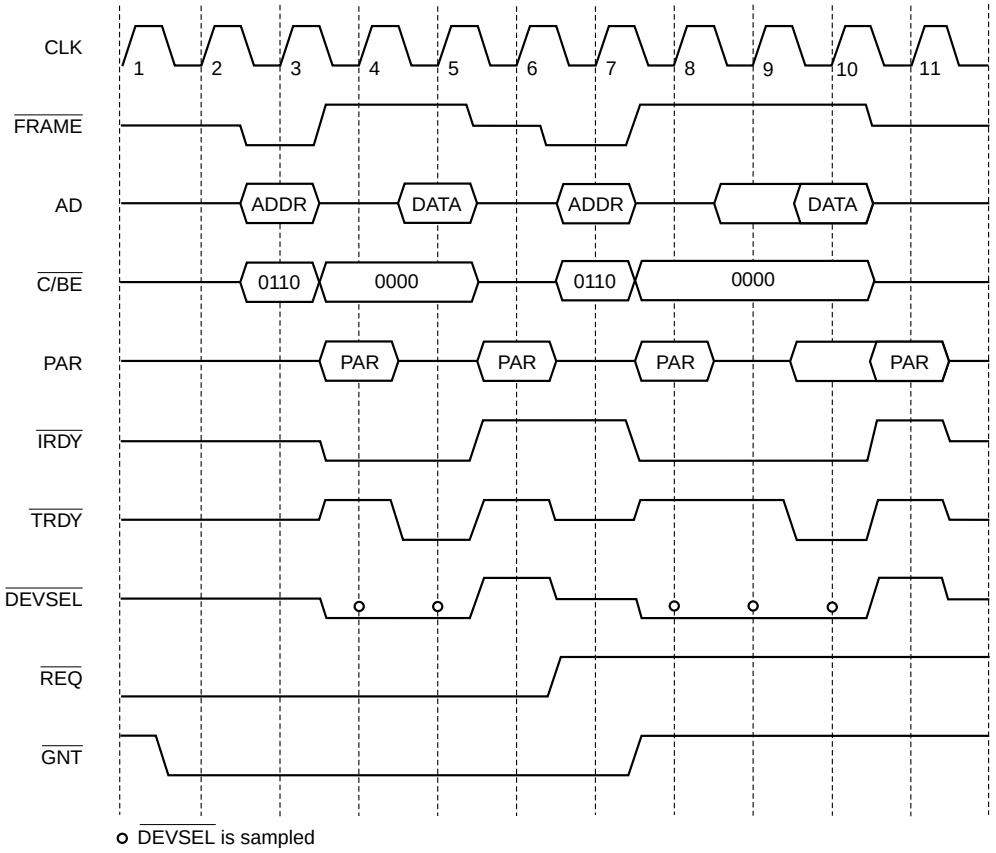


Figure 12. Non-Burst Read Transfer

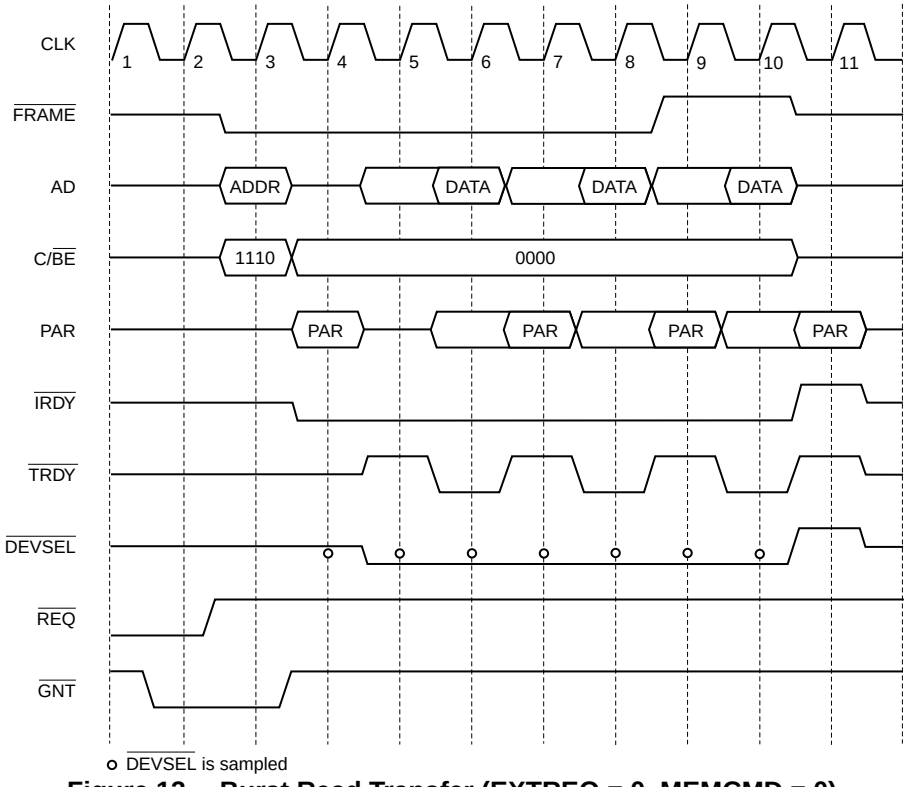


Figure 13. Burst Read Transfer (EXTREQ = 0, MEMCMD = 0)

Basic Non-Burst Write Transfer

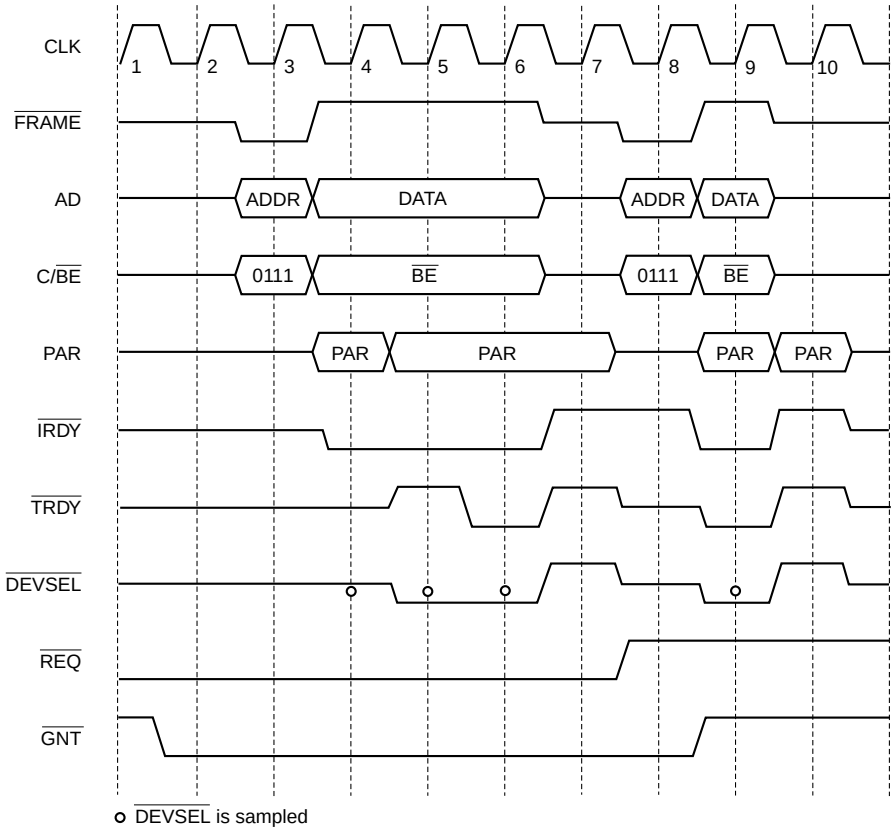
By default, the Am79C972 controller uses non-burst cycles in all bus master write operations. All Am79C972 controller non-burst write accesses are of the PCI command type Memory Write (type 7). The byte enable signals indicate the byte lanes that have valid data.The Am79C972 controller typically performs more than one non-burst write transaction within a single bus mastership period. FRAME is dropped between consecutive non-burst write cycles. REQ, however, stays asserted until FRAME is asserted for the last transaction. The Am79C972 supports zero wait state write cycles except with descriptor write transfers. (See the section *Descriptor DMA Transfers* for the only exception.) It asserts IRDY immediately after the address phase.

Figure 14 shows two non-burst write transactions. The first transaction has two wait states. The target inserts one wait state by asserting DEVSEL one clock late and another wait state by also asserting TRDY one clock late. The second transaction shows a zero wait state write cycle. The target asserts DEVSEL and TRDY in the same cycle as the Am79C972 controller asserts IRDY.

Basic Burst Write Transfer

The Am79C972 controller supports burst mode for all bus master write operations. The burst mode must be enabled by setting BWRITE (BCR18, bit 5). To allow burst transfers in descriptor write operations, the Am79C972 controller must also be programmed to use SWSTYLE 3 (BCR20, bits 7-0). All Am79C972 controller burst write transfers are of the PCI command type Memory Write (type 7). AD[1:0] will both be 0 during the address phase indicating a linear burst order. The byte enable signals indicate the byte lanes that have valid data.

The Am79C972 controller will always perform a single burst write transaction per bus mastership period, where transaction is defined as one address phase and one or multiple data phases. The Am79C972 controller supports zero wait state write cycles except with the case of descriptor write transfers. (See the section *Descriptor DMA Transfers* for the only exception.) The device asserts IRDY immediately after the address phase and at the same time starts sampling DEVSEL. FRAME is deasserted when the next to last data phase is completed.



21485C-17

Figure 14. Non-Burst Write Transfer

Figure 15 shows a typical burst write access. The Am79C972 controller arbitrates for the bus, is granted access, and writes four 32-bit words (DWords) to the 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 TRDY. The example assumes that EXTREQ (BCR18, bit 8) is set to 1, therefore, REQ is not deasserted until the next to last data phase is finished.

Target Initiated Termination

When the Am79C972 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, disconnect without data transfer, and target abort.

Disconnect With Data Transfer

Figure 16 shows a disconnection in which one last data transfer occurs after the target asserted STOP. STOP is asserted on clock 4 to start the termination sequence. Data is still transferred during this cycle, since both IRDY and TRDY are asserted. The Am79C972 controller terminates the current transfer with the deassertion of FRAME on clock 5 and of IRDY one clock later. It finally releases the bus on clock 7. The Am79C972 controller will again request the bus after two clock cycles, if it wants to transfer more data. The starting address of the new transfer will be the address of the next non-transferred data.

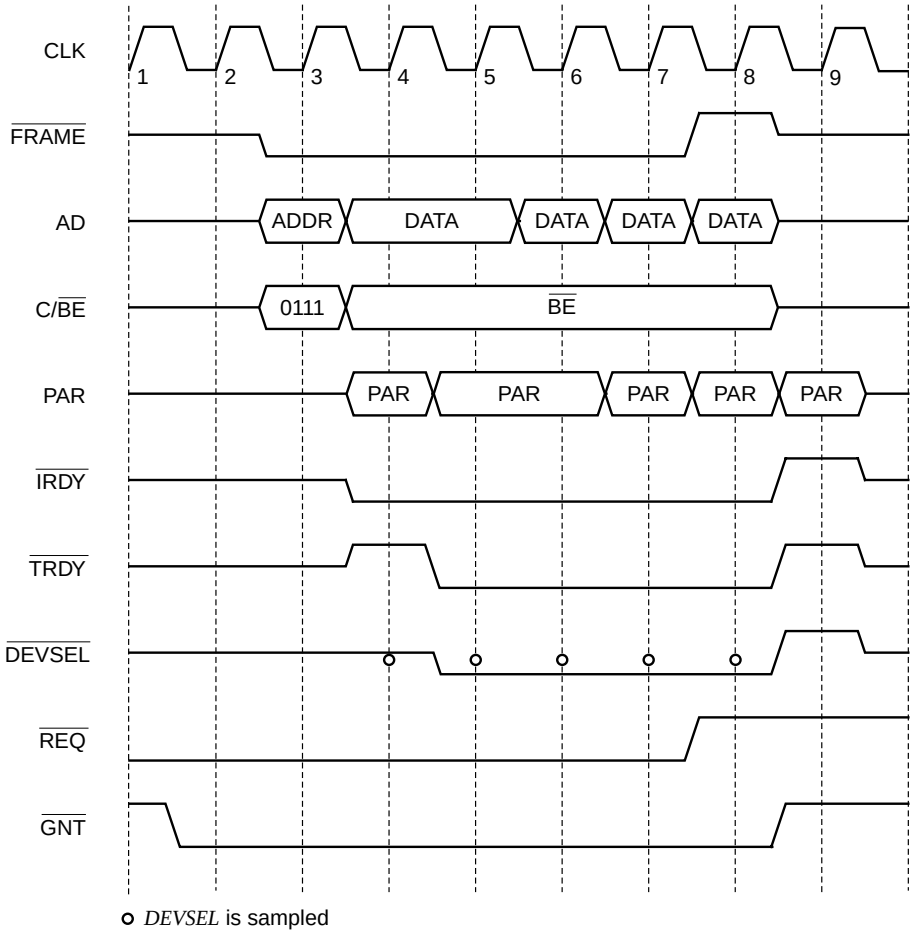


Figure 15. Burst Write Transfer (EXTREQ = 1)

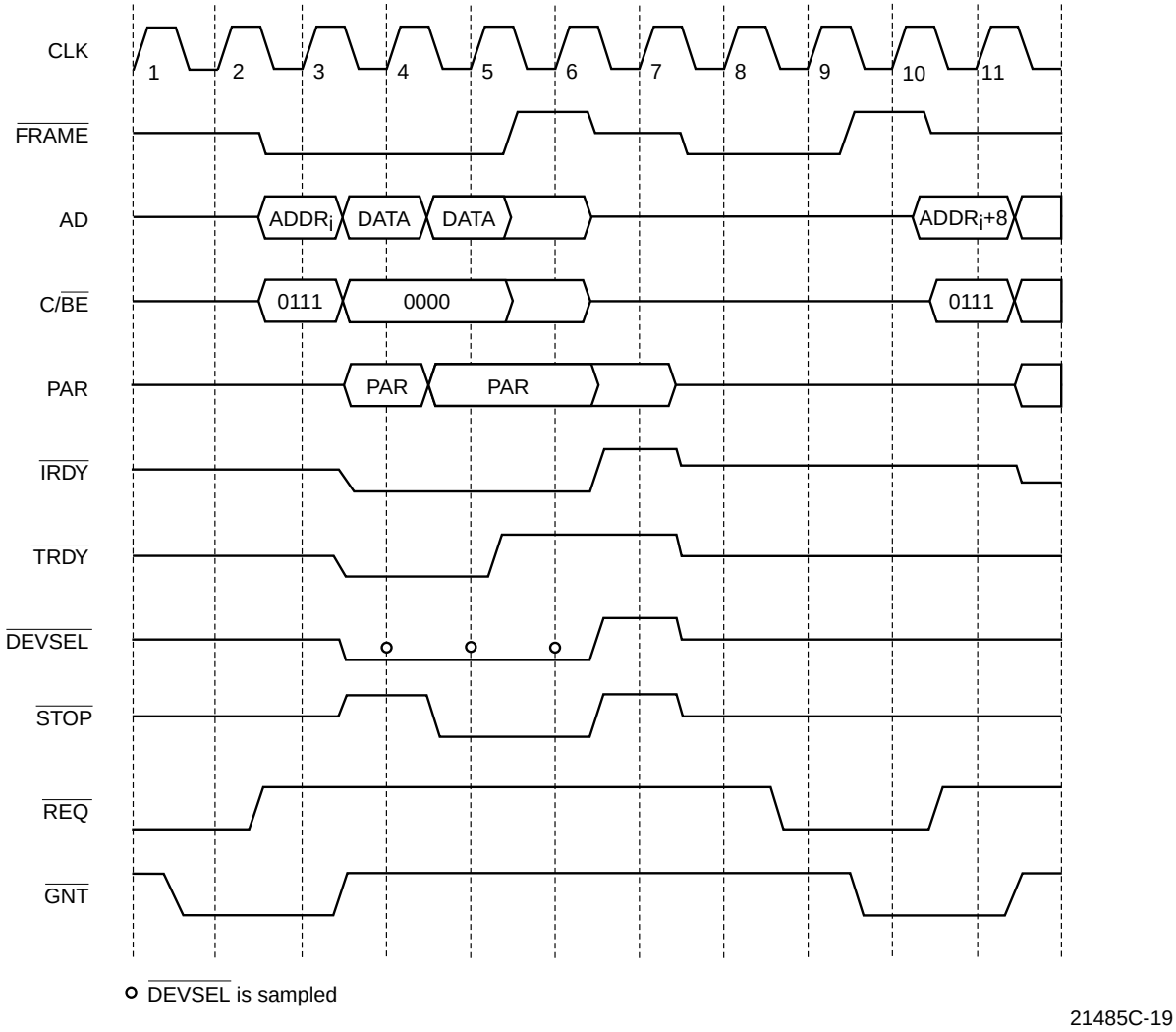


Figure 16. Disconnect With Data Transfer

Disconnect Without Data Transfer

Figure 17 shows a target disconnect sequence during which no data is transferred. STOP is asserted on clock 4 without TRDY being asserted at the same time. The Am79C972 controller terminates the access with the deassertion of FRAME on clock 5 and of IRDY one clock cycle later. It finally releases the bus on clock 7. The Am79C972 controller will again request the bus after two clock cycles to retry the last transfer. The starting address of the new transfer will be the address of the last non-transferred data.

Target Abort

Figure 18 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 Am79C972 controller cannot make any assumption about the success of the previous data transfers in the current transaction. The Am79C972 controller terminates the current transfer with the deassertion of FRAME on clock 5 and of IRDY one clock cycle later. It finally releases the bus on clock 6.

Since data integrity is not guaranteed, the Am79C972 controller cannot recover from a target abort event. The Am79C972 controller will reset all CSR locations to their STOP_RESET values. The BCR and PCI configuration registers will not be cleared. Any on-going network transmission is terminated in an orderly sequence. If less than 512 bits have been transmitted onto the network, the transmission will be terminated immediately, generating a runt packet. If 512 bits or more have been transmitted, the message will have the current FCS inverted and appended at the next byte boundary to guarantee an FCS error is detected at the receiving station.

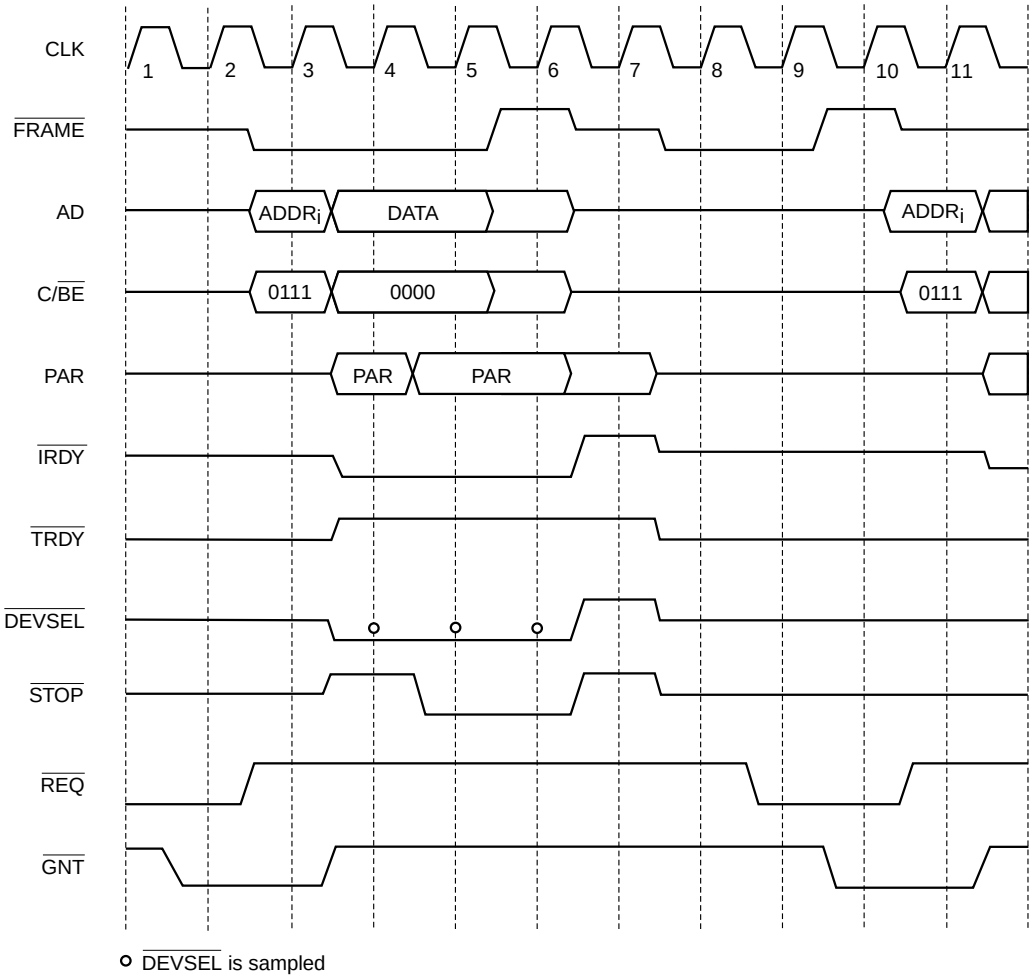


Figure 17. Disconnect Without Data Transfer

RTABORT (PCI Status register, bit 12) will be set to indicate that the Am79C972 controller has received a target abort. In addition, SINT (CSR5, bit 11) will be set to 1. When SINT is set, INTA is asserted if the enable bit SINTE (CSR5, bit 10) is set to 1. This mechanism can be used to inform the driver of the system error. The host can read the PCI Status register to determine the exact cause of the interrupt.

Master Initiated Termination

There are three scenarios besides normal completion of a transaction where the Am79C972 controller will terminate the cycles it produces on the PCI bus.

Preemption During Non-Burst Transaction

When the Am79C972 controller performs multiple non-burst transactions, it keeps REQ asserted until the assertion of FRAME for the last transaction. When GNT is removed, the Am79C972 controller will finish the current transaction and then release the bus. If it is not the last transaction, REQ will remain asserted to regain bus ownership as soon as possible. See Figure 19.

Preemption During Burst Transaction

When the Am79C972 controller operates in burst mode, it only performs a single transaction per bus mastership period, where *transaction* is defined as one address phase and one or multiple data phases. The central arbiter can remove GNT at any time during the transaction. The Am79C972 controller will ignore the deassertion of GNT and continue with data transfers, as long as the PCI Latency Timer is not expired. When the Latency Timer is 0 and GNT is deasserted, the Am79C972 controller will finish the current data phase, deassert FRAME, finish the last data phase, and release the bus. If EXTREQ (BCR18, bit 8) is cleared to 0, it will immediately assert REQ to regain bus ownership as soon as possible. If EXTREQ is set to 1, REQ will stay asserted.

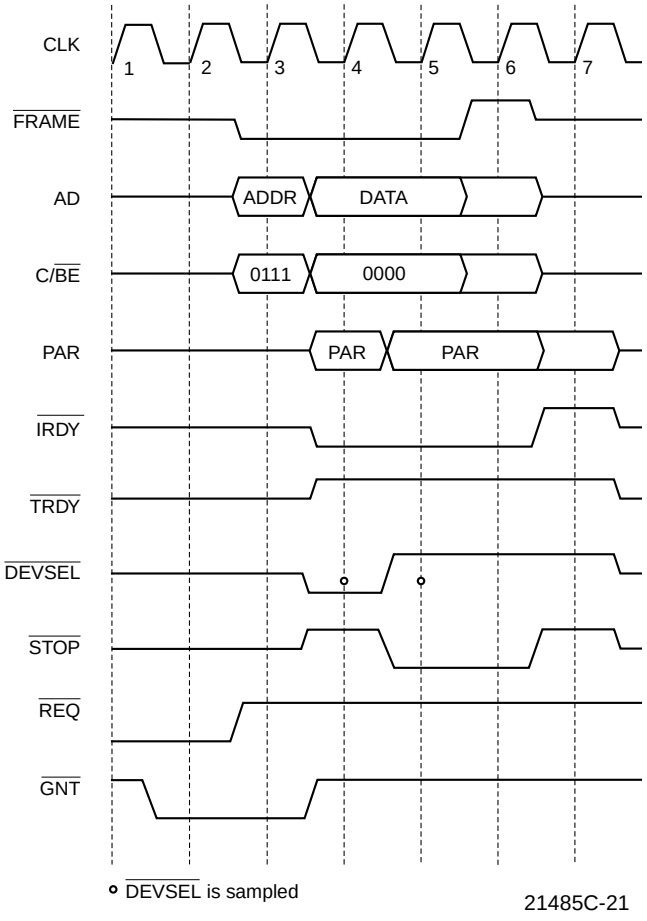


Figure 18. Target Abort

When the preemption occurs after the counter has counted down to 0, the Am79C972 controller will finish the current data phase, deassert FRAME, finish the last data phase, and release the bus. Note that it is important for the host to program the PCI Latency Timer according to the bus bandwidth requirement of the Am79C972 controller. The host can determine this bus bandwidth requirement by reading the PCI MAX_LAT and MIN_GNT registers.

Figure 20 assumes that the PCI Latency Timer has counted down to 0 on clock 7.

Master Abort

The Am79C972 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 Am79C972 controller.

The Am79C972 controller will reset all CSR locations to their STOP_RESET values. The BCR and PCI configuration registers will not be cleared. Any on-going network transmission is terminated in an orderly sequence. If less than 512 bits have been transmitted onto the network, the transmission will be terminated immediately, generating a runt packet. If 512 bits or more have been transmitted, the message will have the current FCS inverted and appended at the next byte boundary to guarantee an FCS error is detected at the receiving station.

RMABORT (in the PCI Status register, bit 13) will be set to indicate that the Am79C972 controller has terminated its transaction with a master abort. In addition, SINT (CSR5, bit 11) will be set to 1. When SINT is set, INTA is asserted if the enable bit SINTE (CSR5, bit 10) is set to 1. This mechanism can be used to inform the driver of the system error. The host can read the PCI Status register to determine the exact cause of the interrupt. See Figure 21.

Parity Error Response

During every data phase of a DMA read operation, when the target indicates that the data is valid by asserting TRDY, the Am79C972 controller samples the AD[31:0], C/BE[3:0] and the PAR lines for a data parity error. When it detects a data parity error, the controller sets PERR (PCI Status register, bit 15) to 1. When reporting of that error is enabled by setting PERREN (PCI Command register, bit 6) to 1, the Am79C972 controller also drives the PERR signal low and sets DATAPERR (PCI Status register, bit 8) to 1. The assertion of PERR follows the corrupted data/byte enables by two clock cycles and PAR by one clock cycle.

Figure 22 shows a transaction that has a parity error in the data phase. The Am79C972 controller asserts PERR on clock 8, two clock cycles after data is valid. The data on clock 5 is not checked for parity, since on a read access PAR is only required to be valid one clock after the target has asserted TRDY. The Am79C972 controller then drives PERR high for one clock cycle, since PERR is a sustained tri-state signal.

During every data phase of a DMA write operation, the Am79C972 controller checks the PERR input to see if the target reports a parity error. When it sees the PERR input asserted, the controller sets PERR (PCI Status register, bit 15) to 1. When PERREN (PCI Command register, bit 6) is set to 1, the Am79C972 controller also sets DATAPERR (PCI Status register, bit 8) to 1.

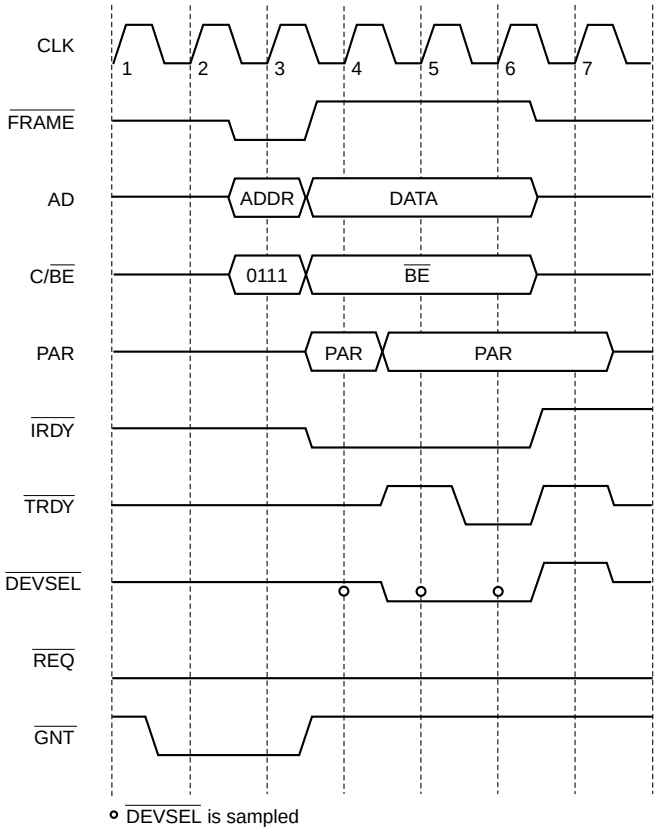


Figure 19. Preemption During Non-Burst Transaction

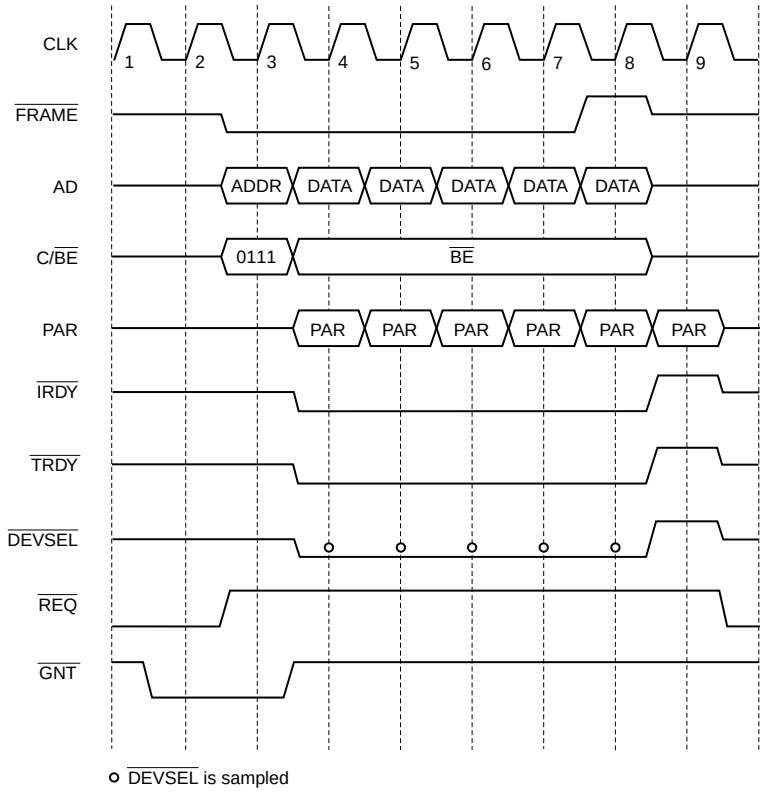
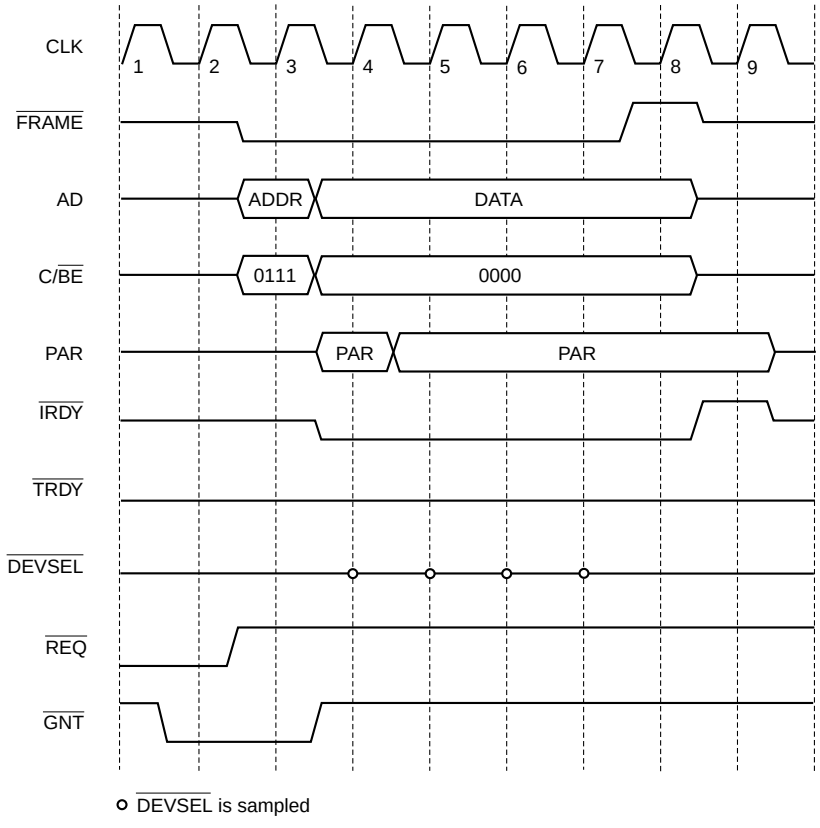
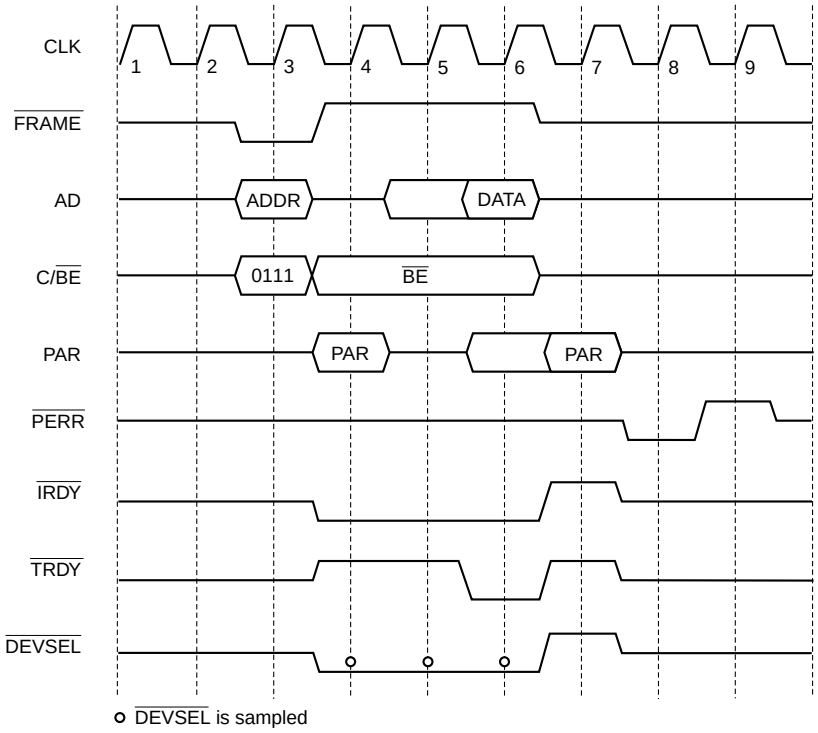


Figure 20. Preemption During Burst Transaction



21485C-24

Figure 21. Master Abort



21485C-25

Figure 22. Master Cycle Data Parity Error Response

Whenever the Am79C972 controller is the current bus master and a data parity error occurs, SINT (CSR5, bit 11) will be set to 1. When SINT is set, INTA is asserted if the enable bit SINTE (CSR5, bit 10) is set to 1. This mechanism can be used to inform the driver of the system error. The host can read the PCI Status register to determine the exact cause of the interrupt. The setting of SINT due to a data parity error is not dependent on the setting of PERREN (PCI Command register, bit 6).

By default, a data parity error does not affect the state of the MAC engine. The Am79C972 controller treats the data in all bus master transfers that have a parity error as if nothing has happened. All network activity continues.

Advanced Parity Error Handling

For all DMA cycles, the Am79C972 controller provides a second, more advanced level of parity error handling. This mode is enabled by setting APERREN (BCR20, bit 10) to 1. When APERREN is set to 1, the BPE bits (RMD1 and TMD1, bit 23) are used to indicate parity error in data transfers to the receive and transmit buffers. Note that since the advanced parity error handling uses an additional bit in the descriptor, SWSTYLE (BCR20, bits 7-0) must be set to 2 or 3 to program the Am79C972 controller to use 32-bit software structures. The Am79C972 controller will react in the following way when a data parity error occurs:

- Initialization block read: STOP (CSR0, bit 2) is set to 1 and causes a STOP_RESET of the device.
- Descriptor ring read: Any on-going network activity is terminated in an orderly sequence and then STOP (CSR0, bit 2) is set to 1 to cause a STOP_RESET of the device.
- Descriptor ring write: Any on-going network activity is terminated in an orderly sequence and then STOP (CSR0, bit 2) is set to 1 to cause a STOP_RESET of the device.
- Transmit buffer read: BPE (TMD1, bit 23) is set in the current transmit descriptor. Any on-going network transmission is terminated in an orderly sequence.
- Receive buffer write: BPE (RMD1, bit 23) is set in the last receive descriptor associated with the frame.

Terminating on-going network transmission in an orderly sequence means that if less than 512 bits have been transmitted onto the network, the transmission will be terminated immediately, generating a runt packet.

If 512 bits or more have been transmitted, the message will have the current FCS inverted and appended at the next byte boundary to guarantee an FCS error is detected at the receiving station.

APERREN does not affect the reporting of address parity errors or data parity errors that occur when the Am79C972 controller is the target of the transfer.

Initialization Block DMA Transfers

During execution of the Am79C972 controller bus master initialization procedure, the Am79C972 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 reading one DWord per transfer and then it will relinquish the bus. When SSIZE32 (BCR20, bit 8) is set to 1 (i.e., the initialization block is organized as 32-bit software structures), there are seven 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 is cleared to 0 (i.e., the initialization block is organized as 16-bit software structures), then three bus mastership periods are needed to complete the initialization sequence.

The Am79C972 supports two transfer modes for reading the initialization block: non-burst and burst mode, with burst mode being the preferred mode when the Am79C972 controller is used in a PCI bus application. See Figure 23 and Figure 24.

When BREADE is cleared to 0 (BCR18, bit 6), all initialization block read transfers will be executed in non-burst mode. There is a new address phase for every data phase. FRAME will be dropped between the two transfers. The two phases within a bus mastership period will have addresses of ascending contiguous order.

When BREADE is set to 1 (BCR18, bit 6), all initialization block read transfers will be executed in burst mode. AD[1:0] will be 0 during the address phase indicating a linear burst order.

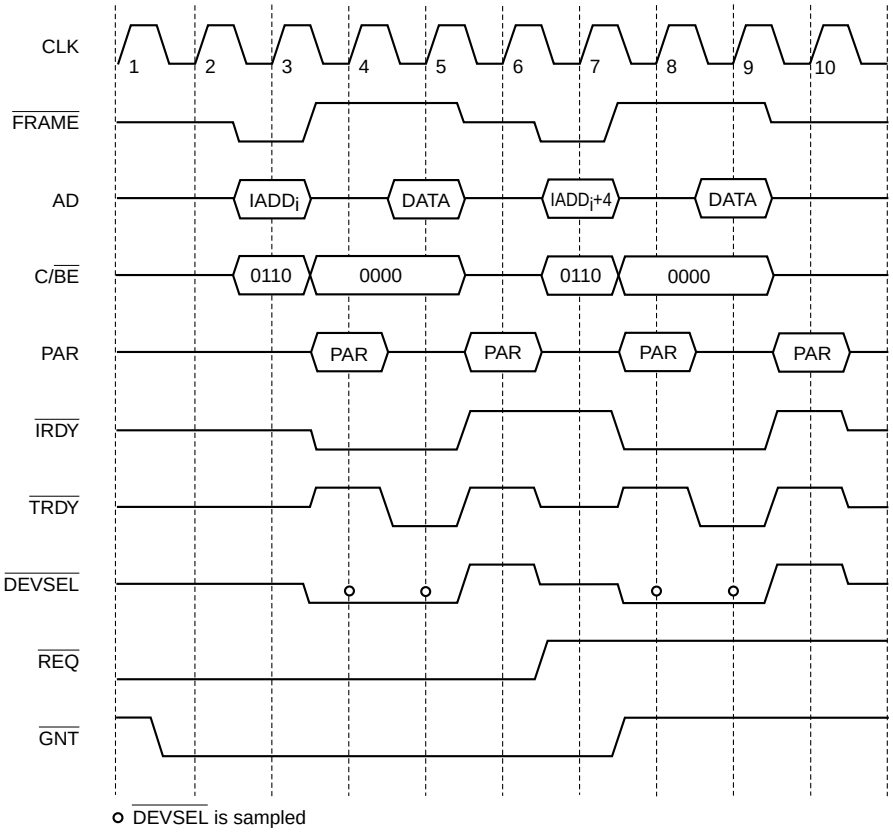


Figure 23. Initialization Block Read In Non-Burst Mode

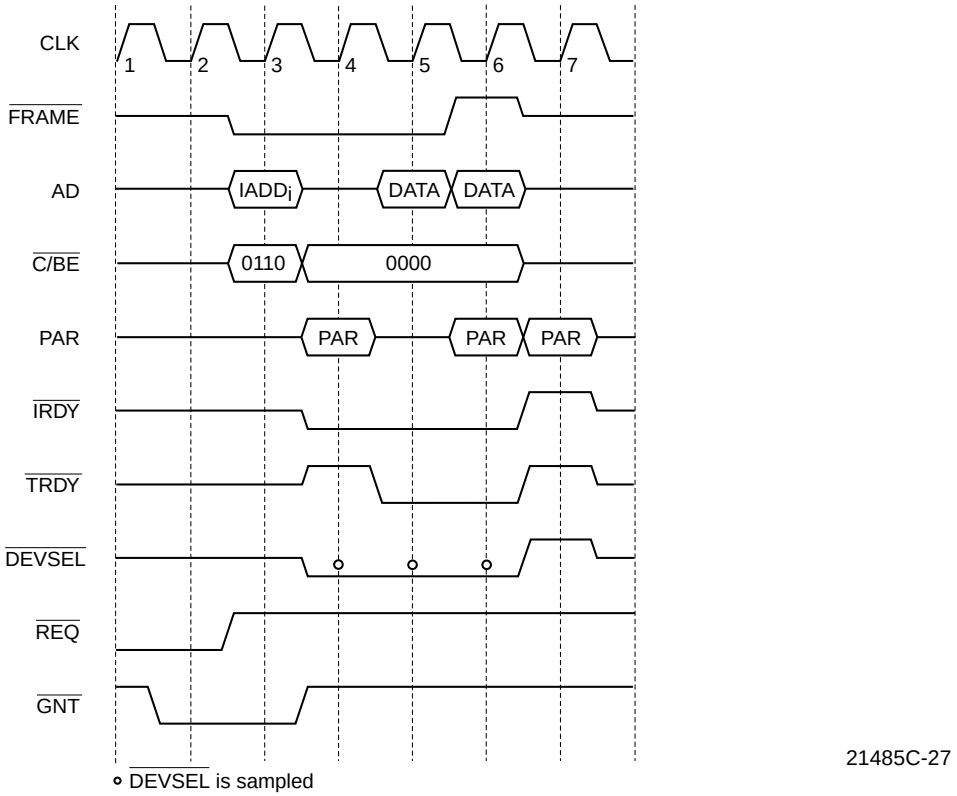


Figure 24. Initialization Block Read In Burst Mode

Descriptor DMA Transfers

Am79C972 microcode will determine when a descriptor access is required. A descriptor DMA read will consist of two data transfers. A descriptor DMA write will consist of one or two data transfers. The descriptor DMA transfers within a single bus mastership period will always be of the same type (either all read or all write).

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 Am79C972 controller will internally discard the extraneous information that was gathered during such a read.

The settings of SWSTYLE (BCR20, bits 7-0) and BREADE (BCR18, bit 6) affect the way the Am79C972 controller performs descriptor read operations.

When SWSTYLE is set to 0 or 2, all descriptor read operations are performed in non-burst mode. The setting of BREADE has no effect in this configuration. See Figure 25.

When SWSTYLE is set to 3, the descriptor entries are ordered to allow burst transfers. The Am79C972 controller will perform all descriptor read operations in burst mode, if BREADE is set to 1. See Figure 26.

Table 4 shows the descriptor read sequence.

During descriptor write accesses, only the byte lanes which need to be written are enabled.

If buffer chaining is used, accesses to the descriptors of all intermediate buffers consist of only one data transfer to return ownership of the buffer to the system. When SWSTYLE (BCR20, bits 7-0) is cleared to 0 (i.e., the descriptor entries are organized as 16-bit software structures), the descriptor access will write a single byte. When SWSTYLE (BCR20, bits 7-0) is set to 2 or 3 (i.e., the descriptor entries are organized as 32-bit software structures), the descriptor access will write a single word. On all single buffer transmit or receive descriptors, as well as on the last buffer in chain, writes to the descriptor consist of two data transfers.

The first data transfer writes a DWord containing status information. The second data transfer writes a byte (SWSTYLE cleared to 0), or otherwise a word containing additional status and the ownership bit (i.e., MD1[31]).

The settings of SWSTYLE (BCR20, bits 7-0) and BWRITE (BCR18, bit 5) affect the way the Am79C972 controller performs descriptor write operations.

When SWSTYLE is set to 0 or 2, all descriptor write operations are performed in non-burst mode. The setting of BWRITE has no effect in this configuration.

When SWSTYLE is set to 3, the descriptor entries are ordered to allow burst transfers. The Am79C972 controller will perform all descriptor write operations in burst mode, if BWRITE is set to 1. See Table 5 for the descriptor write sequence.

A write transaction to the descriptor ring entries is the only case where the Am79C972 controller inserts a wait state when being the bus master. Every data phase in non-burst and burst mode is extended by one clock cycle, during which $\overline{\text{IRDY}}$ is deasserted.

Note that Figure 26 assumes that the Am79C972 controller is programmed to use 32-bit software structures (SWSTYLE = 2 or 3). The byte enable signals for the second data transfer would be 0111b, if the device was programmed to use 16-bit software structures (SWSTYLE = 0).

Table 4. Descriptor Read Sequence

SWSTYLE BCR20[7:0]	BREADE BCR18[6]	AD Bus Sequence
0	X	Address = XXXX XX00h Turn around cycle Data = MD1[31:24], MD0[23:0] Idle Address = XXXX XX04h Turn around cycle Data = MD2[15:0], MD1[15:0]
2	X	Address = XXXX XX04h Turn around cycle Data = MD1[31:0] Idle Address = XXXX XX00h Turn around cycle Data = MD0[31:0]
3	0	Address = XXXX XX04h Turn around cycle Data = MD1[31:0] Idle Address = XXXX XX08h Turn around cycle Data = MD0[31:0]
3	1	Address = XXXX XX04h Turn around cycle Data = MD1[31:0] Data = MD0[31:0]

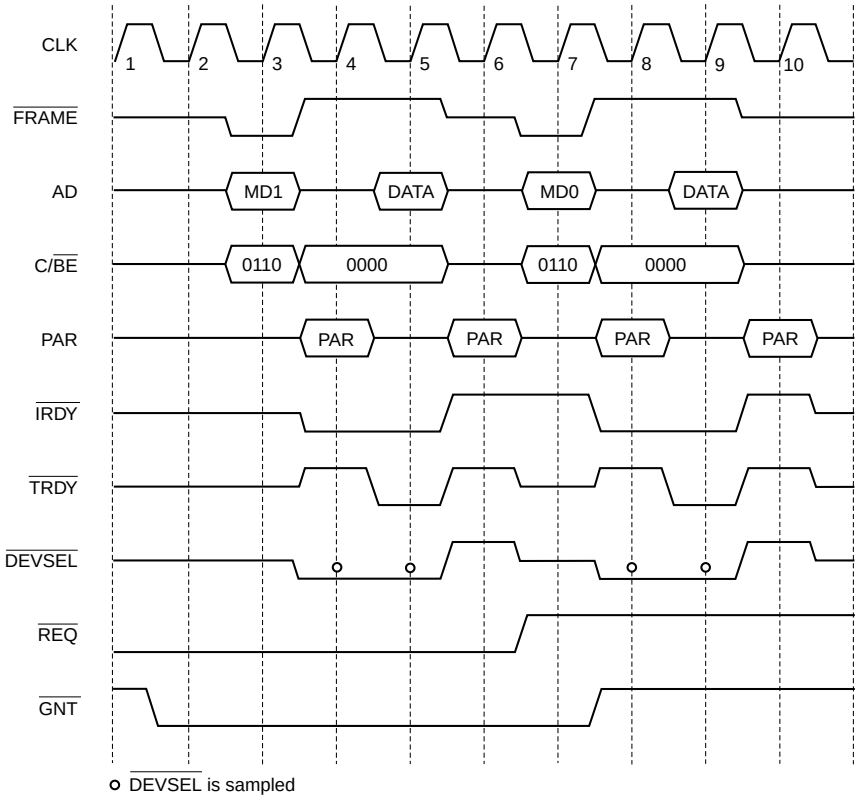


Figure 25. Descriptor Ring Read In Non-Burst Mode

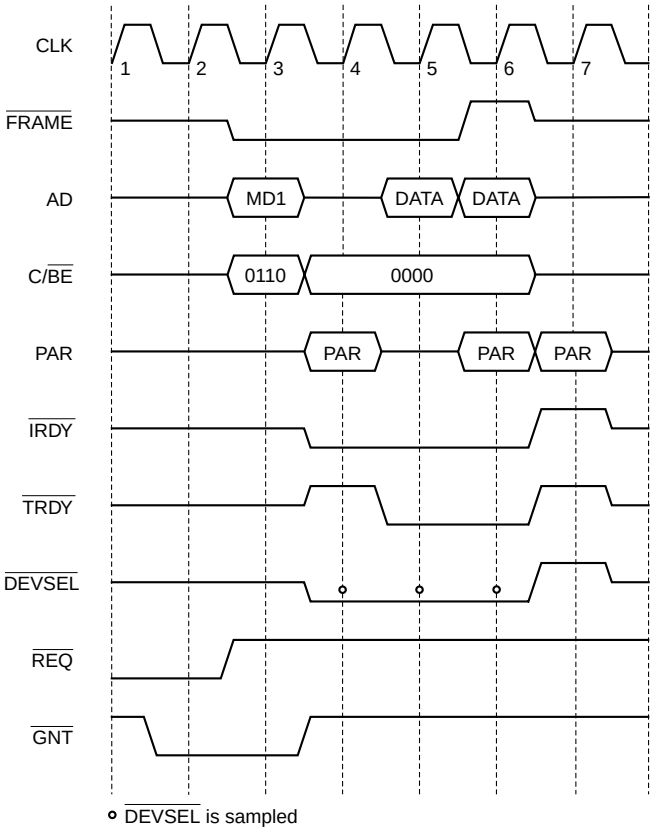


Figure 26. Descriptor Ring Read In Burst Mode

Table 5. Descriptor Write Sequence

SWSTYLE BCR20[7:0]	BWRITE BCR18[5]	AD Bus Sequence
0	X	Address = XXXX XX04h Data = MD2[15:0], MD1[15:0] Idle Address = XXXX XX00h Data = MD1[31:24]
2	X	Address = XXXX XX08h Data = MD2[31:0] Idle Address = XXXX XX04h Data = MD1[31:16]
3	0	Address = XXXX XX00h Data = MD2[31:0] Idle Address = XXXX XX04h Data = MD1[31:16]
3	1	Address = XXXX XX00h Data = MD2[31:0] Data = MD1[31:16]

FIFO DMA Transfers

Am79C972 microcode will determine when a FIFO DMA transfer is required. This transfer mode will be used for transfers of data to and from the Am79C972 FIFOs. Once the Am79C972 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, with burst mode being the preferred mode when the device is used in a PCI bus application.

Non-Burst FIFO DMA Transfers

In the default mode, the Am79C972 controller uses non-burst transfers 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 deasserted after every address phase. Several factors will affect the length of the bus mastership period. The possibilities are as follows:

Bus cycles 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). The exact number of total transfer cycles in the bus mastership period is dependent on all of the following variables: the settings of the FIFO watermarks, the conditions of the FIFOs, the latency of the system bus to the Am79C972 controller's bus request, the speed of bus operation and bus preemption events. 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 $\overline{\text{TRDY}}$ response 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, the slower the bus transfer operations; the slower the clock speed, the higher the transmit watermark; or the higher the receive watermark, the longer the bus mastership period will be.

Note: *The PCI Latency Timer is not significant during non-burst transfers.*

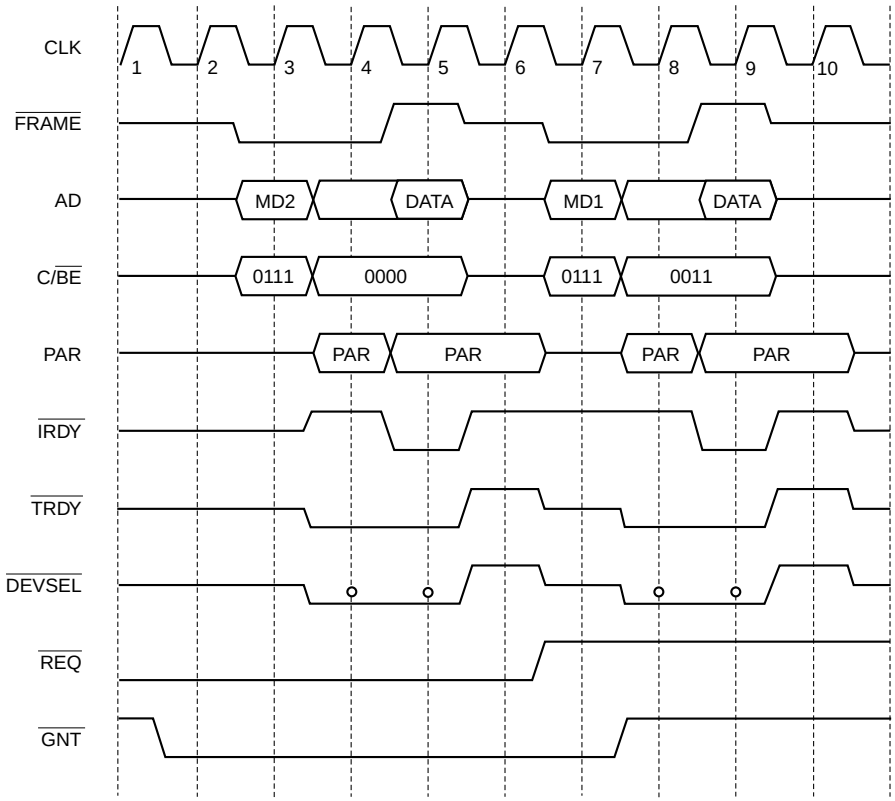


Figure 27. Descriptor Ring Write In Non-Burst Mode

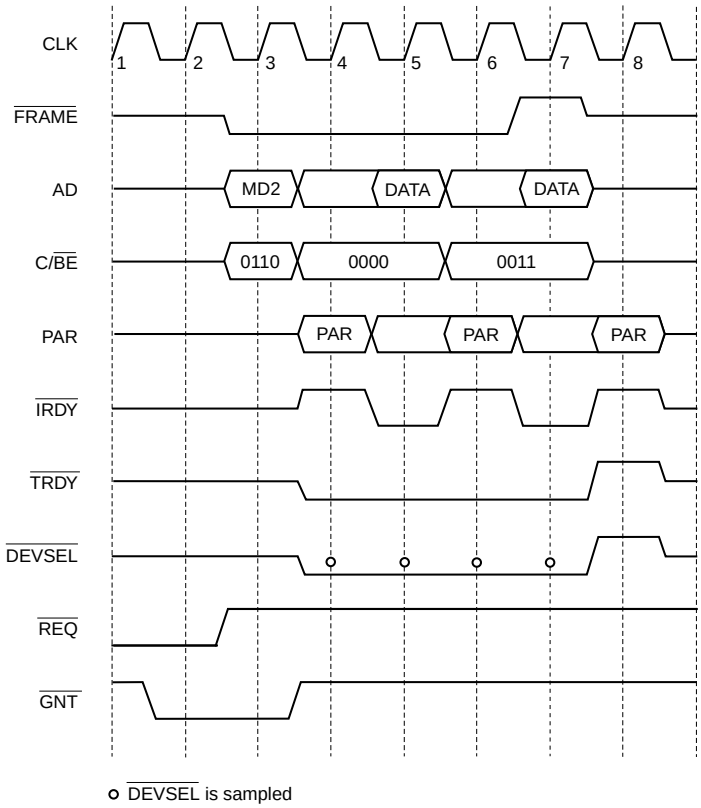


Figure 28. Descriptor Ring Write In Burst Mode

Burst FIFO DMA Transfers

Bursting is only performed by the Am79C972 controller if the BREADE and/or BWRITE bits of BCR18 are set. These bits individually enable/disable the ability of the Am79C972 controller to perform burst accesses during master read operations and master write operations, respectively.

A burst transaction will start with an address phase, followed by one or more data phases. AD[1:0] will always be 0 during the address phase indicating a linear burst order.

During FIFO DMA read operations, all byte lanes will always be active. The Am79C972 controller will internally discard unused bytes. During the first and the last data phases of a FIFO DMA burst write operation, one or more of the byte enable signals may be inactive. All other data phases will always write a complete DWord.

Figure 29 shows the beginning of a FIFO DMA write with the beginning of the buffer not aligned to a DWord boundary. The Am79C972 controller starts off by writing only three bytes during the first data phase. This operation aligns the address for all other data transfers to a 32-bit boundary so that the Am79C972 controller can continue bursting full DWords.

If a receive buffer does not end on a DWord boundary, the Am79C972 controller will perform a non-DWord write on the last transfer to the buffer. Figure 30 shows the final three FIFO DMA transfers to a receive buffer. Since there were only nine bytes of space left in the receive buffer, the Am79C972 controller bursts three data phases. The first two data phases write a full DWord, the last one only writes a single byte.

Note that the Am79C972 controller will always perform a DWord transfer as long as it owns the buffer space, even when there are less than four bytes to write. For example, if there is only one byte left for the current receive frame, the Am79C972 controller will write a full DWord, containing the last byte of the receive frame in the least significant byte position (BSWP is cleared to 0, CSR3, bit 2). The content of the other three bytes is undefined. The message byte count in the receive descriptor always reflects the exact length of the received frame.

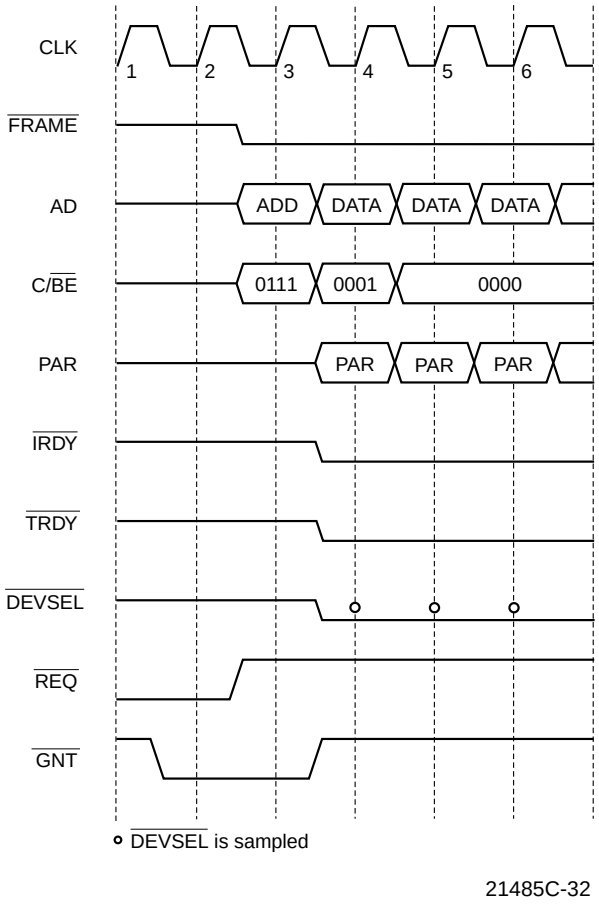


Figure 29. FIFO Burst Write At Start Of Unaligned Buffer

The Am79C972 controller will continue transferring FIFO data 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 the Am79C972 controller is preempted, and the PCI Latency Timer is expired. The host should use the values in the PCI MIN_GNT and MAX_LAT registers to determine the value for the PCI Latency Timer.

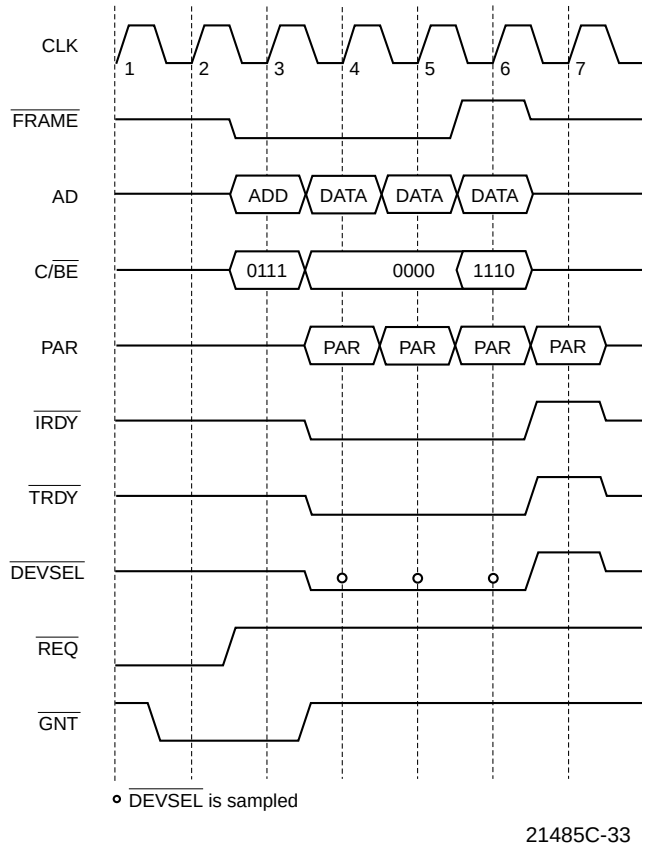


Figure 30. FIFO Burst Write At End Of Unaligned Buffer

The exact number of total transfer cycles in the bus mastership period is dependent on all of the following variables: the settings of the FIFO watermarks, the conditions of the FIFOs, the latency of the system bus to the Am79C972 controller's bus request, and the speed of bus operation. The 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 TRDY response 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, the slower the bus transfer operations; the slower the clock speed, the higher the transmit watermark; or the lower the receive watermark, the longer the total burst length will be.

When a FIFO DMA burst operation is preempted, the Am79C972 controller will not relinquish bus ownership until the PCI Latency Timer expires.

Buffer Management Unit

The Buffer Management Unit (BMU) is a microcoded 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

Am79C972 initialization includes the reading of the initialization block in memory to obtain the operating parameters. The initialization block can be organized in two ways. When SSIZE32 (BCR20, bit 8) is at its default value of 0, all initialization block entries are logically 16-bits wide to be backwards compatible with the Am79C90 C-LANCE and Am79C96x PCnet-ISA family. When SSIZE32 (BCR20, bit 8) is set to 1, all initialization block entries are logically 32-bits wide. Note that the Am79C972 controller always performs 32-bit bus transfers to read the initialization block entries. 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. Once the initialization block has been completely read in and internal registers have been updated, IDON will be set in CSR0, generating an interrupt (if IENA is set).

The Am79C972 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 initialization block contains the user defined conditions for Am79C972 operation, together with the base addresses and length information of the transmit and receive descriptor rings.

There is an alternate method to initialize the Am79C972 controller. Instead of initialization via the initialization block in memory, data can be written directly into the appropriate registers. Either method or a combination of the two may be used at the discretion of the programmer. Please refer to *Appendix A, Alternative Method for Initialization* for details on this alternate method.

Re-Initialization

The transmitter and receiver sections of the Am79C972 controller can be turned on via the initialization block (DTX, DRX, CSR15, bits 1-0). The states of the transmitter and receiver are monitored by the host through CSR0 (RXON, TXON bits). The Am79C972 controller should be re-initialized 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).

Re-initialization 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 Am79C972 controller as in the C-LANCE device. In particular, upon restart, the Am79C972 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 restarting the Am79C972 controller. The reload of descriptor base addresses is performed in the C-LANCE device only after initialization, so that a restart of the C-LANCE without initialization leaves the C-LANCE pointing at the same descriptor locations as before the restart.

Suspend

The Am79C972 controller offers two suspend modes that allow easy updating of the CSR registers without going through a full re-initialization of the device. The suspend modes also allow stopping the device with orderly termination of all network activity.

The host requests the Am79C972 controller to enter the suspend mode by setting SPND (CSR5, bit 0) to 1. The host must poll SPND until it reads back 1 to determine that the Am79C972 controller has entered the suspend mode. When the host sets SPND to 1, the procedure taken by the Am79C972 controller to enter the suspend mode depends on the setting of the fast suspend enable bit (FASTSPND, CSR7, bit 15).

When a fast suspend is requested (FASTSPND is set to 1), the Am79C972 controller performs a quick entry into the suspend mode. At the time the SPND bit is set, the Am79C972 controller will continue the DMA process of any transmit and/or receive packets that have already begun DMA activity until the network activity has been completed. In addition, any transmit packet that had started transmission will be fully transmitted and any receive packet that had begun reception will be fully received. However, no additional packets will be transmitted or received and no additional transmit or receive DMA activity will begin after network activity has ceased. Hence, the Am79C972 controller may enter the suspend mode with transmit and/or receive packets still in the FIFOs or the SRAM. This offers a worst case suspend time of a maximum length packet over the possibility of completely emptying the SRAM. Care must be exercised in this mode, because the entire memory subsystem of the Am79C972 controller is suspended. Any changes to either the descriptor rings or the SRAM can cause the Am79C972 controller to start up in an unknown condition and could cause data corruption.

When FASTSPNDE is 0 and the SPND bit is set, the Am79C972 controller may take longer before entering the suspend mode. At the time the SPND bit is set, the Am79C972 controller will complete the DMA process of a transmit packet if it had already begun and the Am79C972 controller will completely receive a receive packet if it had already begun. The Am79C972 controller will not receive any new packets after the completion of the current reception. Additionally, all transmit packets stored in the transmit FIFOs and the transmit buffer area in the SRAM (if one is present) will be transmitted, and all receive packets stored in the receive FIFOs and the receive buffer area in the SRAM (if selected) will be transferred into system memory. Since the FIFO and the SRAM contents are flushed, it may take much longer before the Am79C972 controller enters the suspend mode. The amount of time that it takes depends on many factors including the size of the SRAM, bus latency, and network traffic level.

Upon completion of the described operations, the Am79C972 controller sets the read-version of SPND to 1 and enters the suspend mode. In suspend mode, all of the CSR and BCR registers are accessible. As long as the Am79C972 controller is not reset while in suspend mode (by H_RESET, S_RESET, or by setting the STOP bit), no re-initialization of the device is required after the device comes out of suspend mode. When SPND is set to 0, the Am79C972 controller will leave the suspend mode and will continue at the transmit and receive descriptor ring locations where it was when it entered the suspend mode.

See the section on *Magic Packet*™ technology for details on how that affects suspension of the Am79C972 controller.

Buffer Management

Buffer management is accomplished through message descriptor entries organized as ring structures in memory. There are two descriptor rings, one for transmit and one for receive. Each descriptor describes a single buffer. A frame may occupy one or more buffers. If multiple buffers are used, this is referred to as buffer chaining.

Descriptor Rings

Each descriptor ring must occupy a contiguous area of memory. During initialization, the user-defined base address for the transmit and receive descriptor rings, as well as the number of entries contained in the descriptor rings are set up. The programming of the software style (SWSTYLE, BCR20, bits 7-0) affects the way the descriptor rings and their entries are arranged.

When SWSTYLE is at its default value of 0, the descriptor rings are backwards compatible with the Am79C90 C-LANCE and the Am79C96x PCnet-ISA family. The descriptor ring base addresses must be aligned to an 8-byte boundary and a maximum of 128 ring entries is allowed when the ring length is set through the TLEN and RLEN fields of the initialization block. Each ring entry contains a subset of the three 32-bit transmit or receive message descriptors (TMD, RMD) that are organized as four 16-bit structures

(SSIZE32 (BCR20, bit 8) is set to 0). Note that even though the Am79C972 controller treats the descriptor entries as 16-bit structures, it will always perform 32-bit bus transfers to access the descriptor entries. The value of CSR2, bits 15-8, is used as the upper 8-bits for all memory addresses during bus master transfers.

When SWSTYLE is set to 2 or 3, the descriptor ring base addresses must be aligned to a 16-byte boundary, and a maximum of 512 ring entries is allowed when the ring length is set through the TLEN and RLEN fields of the initialization block. Each ring entry is organized as three 32-bit message descriptors (SSIZE32 (BCR20, bit 8) is set to 1). The fourth DWord is reserved. When SWSTYLE is set to 3, the order of the message descriptors is optimized to allow read and write access in burst mode.

For any software style, 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:

- The address of the actual message data buffer in user or host memory
- The length of the message buffer
- 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 Am79C972 controller or the host. The OWN bit within the descriptor status information, either TMD or RMD, is used for this purpose.

When OWN is set to 1, it signifies that the Am79C972 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 Am79C972 controller, then the software must not read ahead to the next descriptor. The software should wait at a descriptor it does not own until the Am79C972 controller sets OWN to 0 to release ownership to the software. (When LAPPEN (CSR3, bit 5) is set to 1, this rule is modified. See the LAPPEN description. At initialization, the Am79C972 controller reads the base address of both the transmit and receive descriptor rings into CSRs for use by the Am79C972 controller during subsequent operations.

Figure 31 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, when SSIZE32 is cleared to 0.

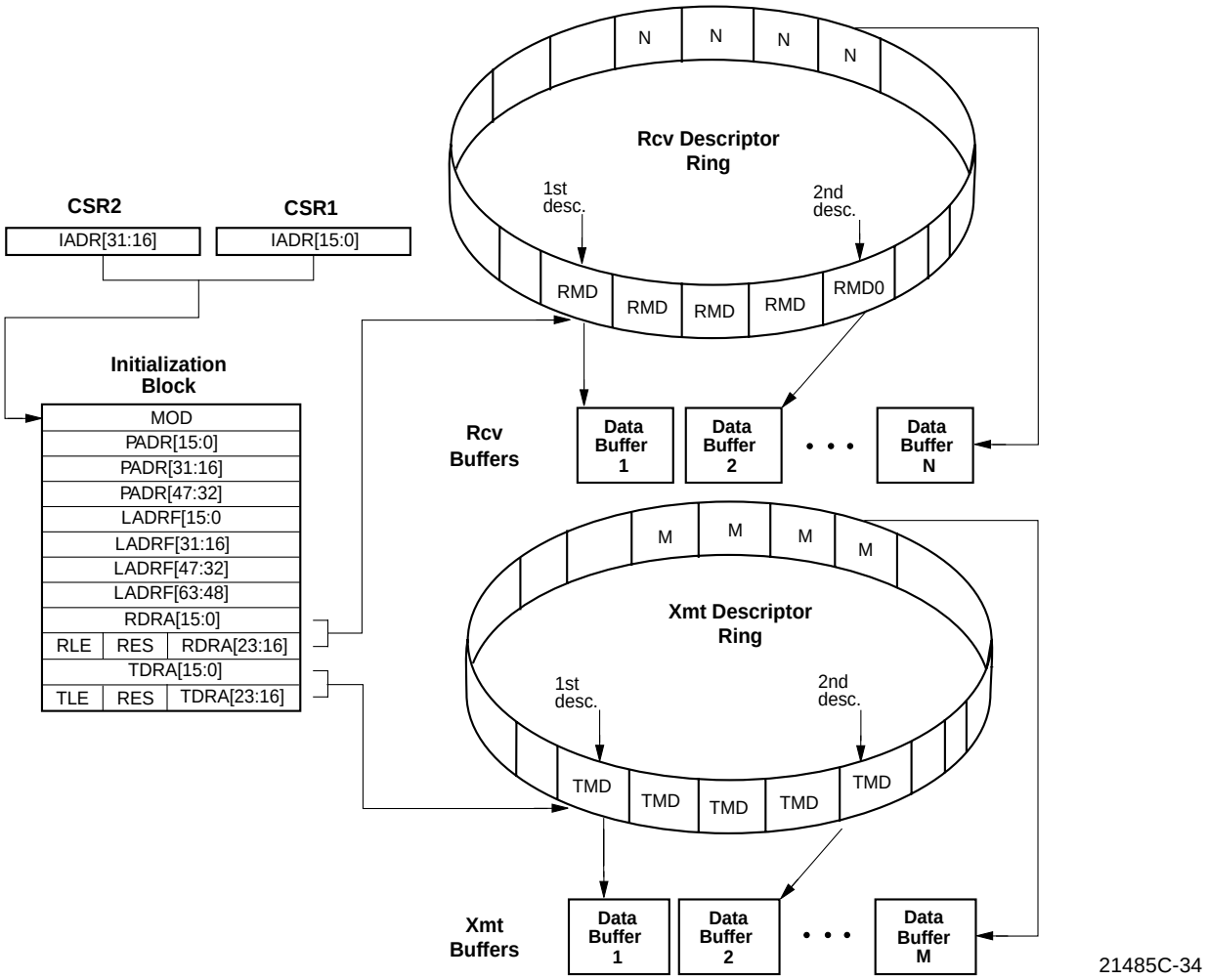


Figure 31. 16-Bit Software Model

Note that the value of CSR2, bits 15-8, is used as the upper 8-bits for all memory addresses during bus master transfers.

Figure 32 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, when SSIZE32 is set to 1.

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 Am79C972 controller, then the Am79C972 controller will periodically poll the current receive and transmit descriptor entries in order to ascertain their ownership. If the TXDPOLL bit in CSR4 is set, then the transmit polling function is disabled.

A typical polling operation consists of the following sequence. The Am79C972 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, if the OWN bit is set (i.e., CSR18, CSR19, CSR20, CSR21, CSR40, CSR42, CSR50, CSR52).

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

1. Am79C972 controller does not own the current RDTE *and* the poll time has elapsed *and* RXON = 1 (CSR0, bit 5), *or*
2. Am79C972 controller does not own the next RDTE *and* there is more than one receive descriptor in the ring *and* the poll time has elapsed *and* RXON = 1.

.

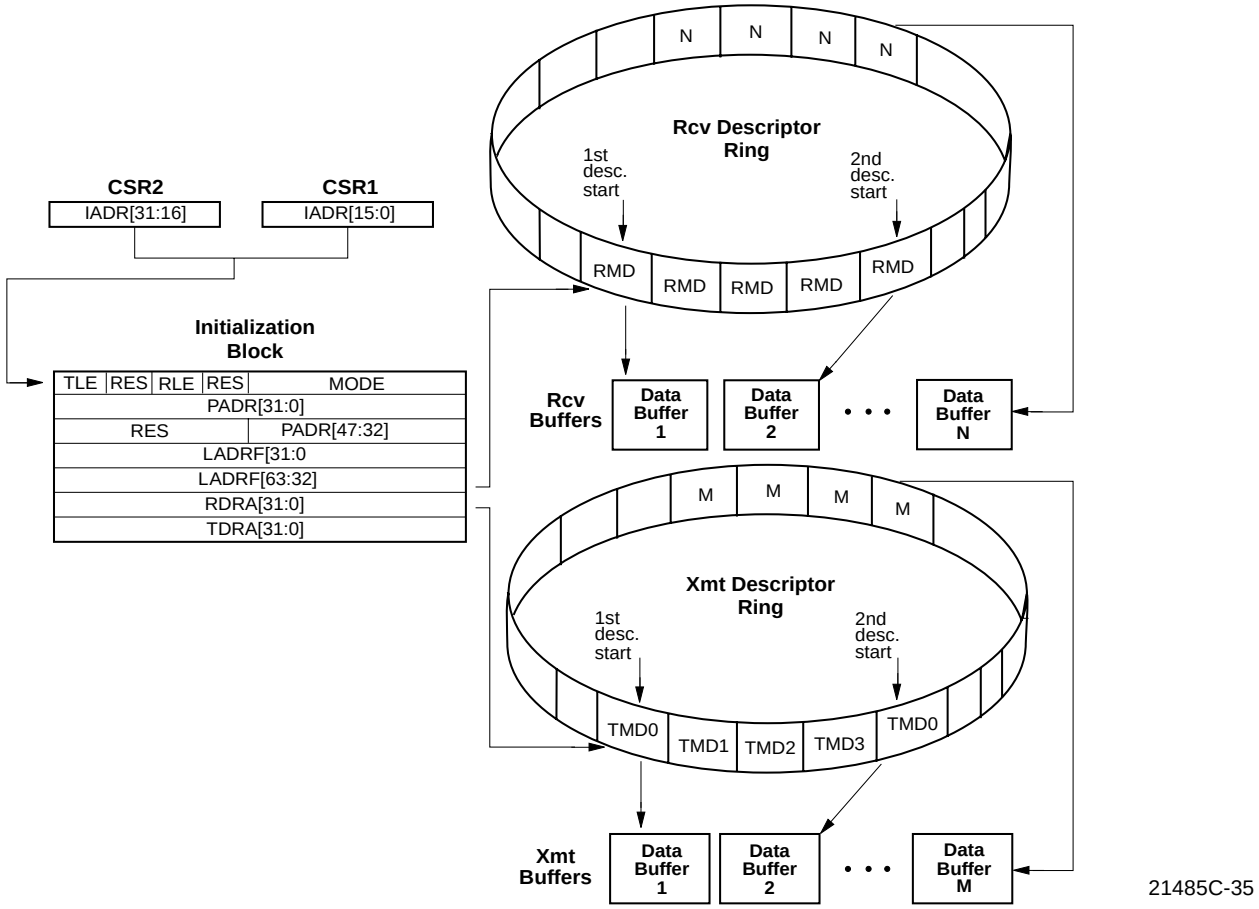


Figure 32. 32-Bit Software Model

If RXON is cleared to 0, the Am79C972 controller will never poll RDTE locations.

In order to avoid missing frames, the system should have at least one RDTE available. To minimize poll activity, two RDTEs should be available. In this case, the poll operation will only consist of the check of the status of the current TDTE.

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

1. Am79C972 controller does not own the current TDTE *and* TXDPOLL = 0 (CSR4, bit 12) *and* TXON = 1 (CSR0, bit 4) *and* the poll time has elapsed, *or*
2. Am79C972 controller does not own the current TDTE *and* TXDPOLL = 0 *and* TXON = 1 *and* a frame has just been received, or
3. Am79C972 controller does not own the current TDTE *and* TXDPOLL = 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).

Transmit Descriptor Table Entry

If, after a Transmit Descriptor Table Entry (TDTE) access, the Am79C972 controller finds that the OWN bit of that TDTE is not set, the Am79C972 controller resumes the poll time count and re-examines the same TDTE at the next expiration of the poll time count.

If the OWN bit of the TDTE is set, but the Start of Packet (STP) bit is not set, the Am79C972 controller will immediately request the bus in order to clear the OWN bit of this descriptor. (This condition would normally be found following a late collision (LCOL) or retry (RTRY) error that occurred in the middle of a transmit frame chain of buffers.) After resetting the OWN bit of this descriptor, the Am79C972 controller will again im-

mediately 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 cleared. In the C-LANCE device, the buffer length of 0 is interpreted as a 4096-byte buffer. A zero length buffer is acceptable as long as it is not the last buffer in a chain (STP = 0 and ENP = 1).

If the OWN bit and STP are set, then microcode control proceeds to a routine that will enable transmit data transfers to the FIFO. The Am79C972 controller will look ahead to the next transmit descriptor after it has performed at least one transmit data transfer from the first buffer.

If the Am79C972 controller does not own the next TDTE (i.e., the second TDTE for this frame), it will complete transmission of the current buffer and update the status of the current (first) TDTE with the BUFF and UFLO bits being set. If DXSUFLO (CSR3, bit 6) is cleared to 0, the underflow error will cause the transmitter to be disabled (CSR0, TXON = 0). The Am79C972 controller will have to be re-initialized to restore the transmit function. Setting DXSUFLO to 1 enables the Am79C972 controller to gracefully recover from an underflow error. The device will scan the transmit descriptor ring until it finds either the start of a new frame or a TDTE it does not own. To avoid an underflow situation in a chained buffer transmission, the system should always set the transmit chain descriptor own bits in reverse order.

If the Am79C972 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 (clear 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.)

It is imperative that the host system never reads the TDTE OWN bits out of order. The Am79C972 controller normally clears OWN bits in strict FIFO order. However, the Am79C972 controller can queue up to two frames in the transmit FIFO. When the second frame uses buffer chaining, the Am79C972 controller might return ownership out of normal FIFO order. The OWN bit for last (and maybe only) buffer of the first frame is not cleared until transmission is completed. During the transmission the Am79C972 controller will read in buffers for the next frame and clear their OWN bits for all but the last one. The first and all intermediate buffers of the second frame can have their OWN bits cleared before the Am79C972 controller returns ownership for the last buffer of the first frame.

If an error occurs in the transmission before all of the bytes of the current buffer have been transferred, transmit status of the current buffer will be immediately updated. If the buffer does not contain the end of packet, the Am79C972 controller will skip over the rest of the frame which experienced the error. This is done by returning to the polling microcode where the Am79C972 controller will clear the OWN bit for all descriptors with OWN = 1 and STP = 0 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 Am79C972 controller will always perform another polling operation. As described earlier, this polling operation will begin with a check of the current RDTE, unless the Am79C972 controller already owns that descriptor. Then the Am79C972 controller will poll the next TDTE. If the transmit descriptor OWN bit has a 0 value, the Am79C972 controller will resume incrementing the poll time counter. If the transmit descriptor OWN bit has a value of 1, the Am79C972 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 Am79C972 controller to avoid inserting poll time counts between successive transmit frames.

By default, whenever the Am79C972 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 cleared. The Am79C972 controller provides two modes to reduce the number of transmit interrupts. The interrupt of a successfully transmitted frame can be suppressed by setting TINTOKD (CSR5, bit 15) to 1. Another mode, which is enabled by setting LTINTEN (CSR5, bit 14) to 1, allows suppression of interrupts for successful transmissions for all but the last frame in a sequence.

Receive Descriptor Table Entry

If the Am79C972 controller does not own both the current and the next Receive Descriptor Table Entry (RDTE), then the Am79C972 controller will continue to poll according to the polling sequence described above. If the receive descriptor ring length is one, 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 Am79C972 controller, then additional poll accesses are not necessary. Future poll operations will not include RDTE accesses as long as the Am79C972 controller retains ownership of the current and the next RDTE.

When receive activity is present on the channel, the Am79C972 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 Am79C972 controller checks the current receive buffer status register CRST (CSR41) to determine the ownership of the current buffer.

If ownership is lacking, the Am79C972 controller will immediately perform a final poll of the current RDTE. If ownership is still denied, the Am79C972 controller has no buffer in which to store the incoming message. The MISS bit will be set in CSR0 and the Missed Frame Counter (CSR112) will be incremented. Another poll of the current RDTE will not occur until the frame has finished.

If the Am79C972 controller sees that the last poll (either a normal poll, or the final effort described in the above paragraph) of the current RDTE shows valid ownership, 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 Am79C972 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 Am79C972 controller does not own the second buffer, ownership of the current descriptor will be passed back to the system by writing a 0 to the OWN bit of RMD1. 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 Am79C972 controller does own the second (next) buffer, ownership will be passed back to the system by writing a 0 to the OWN bit of RMD1 when the first buffer is full. The OWN bit is the only bit modified in the descriptor. Receive data transfers to the second buffer may occur before the Am79C972 controller proceeds to look ahead to the ownership of the third buffer. Such action will depend upon the state of the FIFO when the OWN bit has been updated in 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.

This activity continues until the Am79C972 controller recognizes the completion of the frame (the last byte of this receive message has been removed from the FIFO). The Am79C972 controller will subsequently update the current RDTE status with the end of frame (ENP) indication set, write the message byte count (MCNT) for the entire frame into RMD2, and overwrite the “current” entries in the CSRs with the “next” entries.

Receive Frame Queuing

The Am79C972 controller supports the lack of RDTEs when SRAM (SRAM SIZE in BCR 25, bits 7-0) is enabled through the Receive Frame Queuing mechanism. When the SRAM SIZE = 0, then the Am79C972 controller reverts back to the PCnet PCI II mode of operation. This operation is automatic and does not require any programming by the host. When SRAM is enabled, the Receive Frame Queuing mechanism allows a slow protocol to manage more frames without the high frame loss rate normally attributed to FIFO based network controllers.

The Am79C972 controller will store the incoming frames in the extended FIFOs until polling takes place; if enabled, it discovers it owns an RDTE. The stored frames are not altered in any way until written out into system buffers. When the receive FIFO overflows, further incoming receive frames will be missed during that time. As soon as the network receive FIFO is empty, incoming frames are processed as normal. Status on a per frame basis is not kept during the overflow process. Statistic counters are maintained and accurate during that time.

During the time that the Receive Frame Queuing mechanism is in operation, the Am79C972 controller relies on the Receive Poll Time Counter (CSR 48) to control the worst case access to the RDTE. The Receive Poll Time Counter is programmed through the Receive Polling Interval (CSR49) register. The Received Polling Interval defaults to approximately 2 ms. The Am79C972 controller will also try to access the RDTE during normal descriptor accesses whether they are transmit or receive accesses. The host can force the Am79C972 controller to immediately access the RDTE by setting the RDMD (CSR 7, bit 13) to 1. Its operation is similar to the transmit one. The polling process can be disabled by setting the RXDPOLL (CSR7, bit 12) bit. This will stop the automatic polling process and the host must set the RDMD bit to initiate the receive process into host memory. Receive frames are still stored even when the receive polling process is disabled.

Software Interrupt Timer

The Am79C972 controller is equipped with a software programmable free-running interrupt timer. The timer is constantly running and will generate an interrupt STINT (CSR 7, bit 11) when STINITE (CSR 7, bit 10) is set to 1. After generating the interrupt, the software timer will load the value stored in STVAL and restart. The timer value STVAL (BCR31, bits 15-0) is interpreted as an unsigned number with a resolution of 256 Time Base Clock periods. For instance, a value of 122 ms would be programmed with a value of 9531 (253Bh), if the Time Base Clock is running at 20 MHz. The default value of STVAL is FFFFh which yields the approximate maximum 838 ms timer duration. A write to STVAL restarts the timer with the new contents of STVAL.

Media Access Control

The Media Access Control (MAC) engine incorporates the essential protocol requirements for operation of an Ethernet/IEEE 802.3-compliant node and provides the interface between the FIFO subsystem and the MII.

This section describes operation of the MAC engine when operating in half-duplex mode. When operating in half-duplex mode, 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). When operating in full-duplex mode, the MAC engine behavior changes as described in the section *Full-Duplex Operation*.

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

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, except in full-duplex operation)
  - — Contention resolution (collision handling, except in full-duplex operation)

Transmit and Receive Message Data Encapsulation

The MAC engine provides minimum frame size enforcement for transmit and receive frames. When APAD_XMT (CSR, bit 11) is set to 1, 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 (CSR4, bit 10) is set to 1, the receiver will automatically strip pad bytes from the received message by observing the value in the length field and by 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

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 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 these fields in the frame. The MAC does not use the content in the length/type field unless APAD_XMT (CSR4, bit 11) is set and the data portion of the frame is shorter than 60 bytes.

During GPSI operation, 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. During MII operation, the MAC engine will detect the incoming preamble sequence when the RX_DV signal is activated by the external PHY. The MAC will discard the preamble and begin searching for the SFD except in the case of 100BASE-T4. In that case, the SFD will be the first nibble across the MII interface. Once the SFD is detected, all subsequent nibbles 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 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, all frame bytes including FCS will be passed unmodified to the receive buffer, regardless of the actual frame length.

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 Am79C972 controller has the ability to accept runt packets for diagnostic purposes and proprietary networks.

Destination Address Handling

The first 6 bytes of information after SFD will be interpreted as the destination address field. The MAC en-

gine provides facilities for physical (unicast), logical (multicast), and broadcast address reception.

Error Detection

The MAC engine provides several facilities which report and recover from errors on the medium. In addition, it protects the network 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 Transmit Message Descriptor (TMD) and Control and Status Register (CSR) areas:

- The number of transmission retry attempts (ONE, MORE, RTRY, and TRC).
- Whether the MAC engine had to Defer (DEF) due to channel activity.
- Excessive deferral (EXDEF), indicating that the transmitter experienced Excessive Deferral on this transmit frame, where Excessive Deferral is defined in the ISO 8802-3 (IEEE/ANSI 802.3) standard.
- 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 first 4 µs after a transmission was completed. This may be due to a failed transceiver, disconnected or faulty transceiver drop cable, or because the transceiver does not support this feature (or it is disabled). SQE Test is only valid for 10-Mbps networks.

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 as an invalid FCS (which will also cause the receiver to reject the message).

The status of each receive message is available in the appropriate Receive Message Descriptor (RMD) and CSR areas. All received frames are passed to the host regardless of any error. The FRAM error will only be reported if an FCS error is detected and there is a non-integral number of bytes in the message.

During the reception, the FCS is generated on every nibble (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 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 framing error is reported to the user as follows:

- If the number of dribbling bits are 1 to 7 and there is no FCS error, then there is no Framing error (FRAM = 0).
- If the number of dribbling bits are 1 to 7 and there is a FCS error, then there is also a Framing error (FRAM = 1).
- If the number of dribbling bits is 0, then there is no Framing error. There may or may not be a FCS error.
- If the number of dribbling bits is EIGHT, then there is no Framing error. FCS error will be reported and the receive message count will indicate one extra byte.

Media Access Management

The basic requirement for all stations on the network is to provide fairness of channel allocation. The IEEE 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) 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 optionally a two-part deferral after a receive message.

See ANSI/IEEE Std 802.3-1993 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 inter-Frame 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 InterFrame-SpacingPart1 is other than 0:

1. *Upon completing a transmission, start timing the interrupted gap, as soon as transmitting and carrier sense are both false.*
2. *When timing an inter-frame gap following reception, reset the inter-frame gap timing if carrier sense becomes true during the first 2/3 of the inter-frame 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 0.*

The MAC engine implements the optional receive two part deferral algorithm, with an InterFrameSpacing-Part1 time of 6.0 µs. The InterFrameSpacingPart 2 interval is, therefore, 3.4 µs.

The Am79C972 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 deasserted. During the first part deferral (InterFrameSpacingPart1 - IFS1), the Am79C972 controller will defer any pending transmit frame and respond to the receive message. The IPG counter will be cleared to 0 continuously until the carrier deasserts, 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 Am79C972 controller will begin timing the second part deferral (InterFrameSpacingPart2 - IFS2) of 3.4 µs. Once IFS1 has completed and IFS2 has commenced, the Am79C972 controller will not defer to a receive frame if a transmit frame is pending. This means that the Am79C972 controller will not attempt to receive the receive frame, since it will start to transmit and generate a collision at 9.6 µs. The Am79C972 controller will complete the preamble (64-bit) and jam (32-bit) sequence before ceasing transmission and invoking the random backoff algorithm.

The Am79C972 controller allows the user to program the IPG and the first part deferral (InterFrame-SpacingPart1 - IFS1) through CSR125. By changing the IPG default value of 96 bit times (60h), the user can adjust the fairness or aggressiveness of the Am79C972 MAC on the network. By programming a lower number of bit times than the ISO/IEC 8802-3 standard requires, the Am79C972 MAC engine will become more aggressive on the network. This aggressive nature will give rise to the Am79C972 controller possibly *capturing the network* at times by forcing other less aggressive compliant nodes to defer. By programming a larger number of bit times, the Am79C972 MAC will become less aggressive on the network and may defer more often than normal. The performance of the Am79C972 controller may decrease as the IPG value is increased from the default value, but the resulting behavior may improve network performance by reducing collisions. The Am79C972 controller uses the same IPG for back-to-back transmits and receive-to-transmit accesses. Changing IFS1 will alter the period for which the Am79C972 MAC engine will defer to incoming receive frames.

CAUTION: Care must be exercised when altering these parameters. Adverse network activity could result!

This transmit two-part deferral algorithm is implemented as an option which can be disabled using the DXMT2PD bit in CSR3. The IFS1 programming will have no effect when DXMT2PD is set to 1, but the IPG programming value is still valid. 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 should generate the SQE Test message within 0.6 to 1.6 µs after the transmission ceases. During the time period in which the SQE Test message is expected, the Am79C972 controller will not respond to receive carrier sense.

See ANSI/IEEE Std 802.3-1993 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 Am79C972 controller implements a carrier sense "blinding" period of 4.0 µs length starting from the deassertion 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 network, and since the fragment size will be larger than the 4 µs blinding window, the IPG counter will be reset by a worst case IPG shrinkage/fragment scenario and the Am79C972 controller will defer its transmission. If carrier is detected within the 4.0 to 6.0 µs IFS1 period, the Am79C972 controller will not restart the "blinding" period, but only restart IFS1.

Collision Handling

Collision detection is performed and reported to the MAC engine via the COL/CLSN input pin.

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 rescheduled to a time determined by the random backoff algorithm. If a single retry was required, the 1 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 (1 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 retransmission 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 retransmission attempt is chosen as a uniformly distributed random integer r in the range:*
>
> $$0 \le r < 2^k \text{ where } k = \min(n,10).”$$

The Am79C972 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, while 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.

This modified backoff algorithm is enabled when EMBA (CSR3, bit 3) is set to 1.

Transmit Operation

The transmit operation and features of the Am79C972 controller are controlled by programmable options. The Am79C972 controller offers a large 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.

The disable FCS generation/transmission feature can be programmed as a static feature or dynamically on a frame-by-frame basis.

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 frame 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 frame data has been placed into the transmit FIFO.) The default value of XMTSP is 01b, 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 IEEE 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 IEEE 802.3 frame. FCS is always added if the frame is padded, regardless of the state of DXMTFCS (CSR15, bit 3) or ADD_FCS (TMD1, bit 29). The transmit frame will be padded by bytes with the value of 00H. The default value of APAD_XMT is 0, which will disable automatic pad generation after H_RESET.

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 frame (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 Am79C972 controller to compute the actual number of pad bytes to be inserted. The Am79C972 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 Am79C972 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. See Figure 33.

.

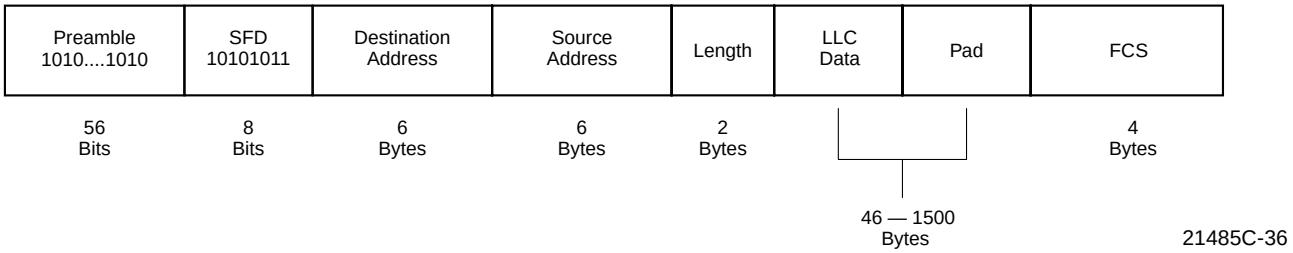


Figure 33. ISO 8802-3 (IEEE/ANSI 802.3) Data Frame

The 544 bit count is derived from the following:

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

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

Preamble/SFD + (Min Frame Size - FCS)

64 + (512-32) = 544 bits

A minimum length transmit frame from the Am79C972 controller, therefore, will be 576 bits, after the FCS is appended.

Transmit FCS Generation

Automatic generation and transmission of FCS for a transmit frame depends on the value of DXMTFCS (CSR15, bit 3). If DXMTFCS is cleared to 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 to frames shorter than 64 bytes by the Am79C972 controller regardless of the state of DXMTFCS or ADD_FCS (TMD1, bit 29). Note that the calculated FCS is transmitted most significant bit first. The default value of DXMTFCS is 0 after H_RESET.

ADD_FCS (TMD1, bit 29) allows the automatic generation and transmission of FCS on a frame-by-frame basis. DXMTFCS should be set to 1 in this mode. To generate FCS for a frame, ADD_FCS must be set in all descriptors of a frame (STP is set to 1). Note that bit 29 of TMD1 has the function of ADD_FCS if SWSTYLE (BCR20, bits 7-0) is programmed to 0, 2, or 3.

Transmit Exception Conditions

Exception conditions for frame transmission fall into two distinct categories: those conditions 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 Am79C972 controller include collisions within the slot time with automatic retry. The Am79C972 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. Note that if DRTY (CSR15, bit 5) is set to 1 or if the network interface is operating in full-duplex mode, no collision handling is required, and any byte of

frame data in the FIFO can be overwritten as soon as it is transmitted.

If 16 total attempts (initial attempt plus 15 retries) fail, the Am79C972 controller sets the RTRY bit in the current transmit TDTE in host memory (TMD2), gives up ownership (resets the OWN bit to 0) 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 100-Mbps networks.)

These conditions should not occur on a correctly configured IEEE 802.3 network operating in half-duplex mode. If they do, they will be reported. None of these conditions will occur on a network operating in full-duplex mode. (See the section *Full-Duplex Operation* for more detail.)

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 cleared until the STP (the next frame) is found.

Loss of Carrier

When operating in half-duplex mode, a loss of carrier condition will be reported if the Am79C972 controller cannot observe receive activity while it is transmitting on the GPSI port.

When the MII port is selected, LCAR will be reported for every frame transmitted if the controller detects a loss of carrier.

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 Am79C972 controller will abandon the transmit process for that 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 retried. Recovery from this condition must be performed by upper layer software.

SQE Test Error

In GPSI mode, CLSN must be asserted after the transmission or otherwise CERR will be set. CERR will be asserted in the 10BASE-T mode through the MII after transmit, if the network port 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 Am79C972 controller are controlled by programmable options. The Am79C972 controller offers a large receive FIFO to provide frame buffering for increased system latency, automatic flushing of collision fragments (runt packets), automatic receive pad stripping, and a variety of address match options.

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 IEEE 802.3 frame format.

All receive frames can be accepted by setting the PROM bit in CSR15. Acceptance of unicast and broadcast frames can be individually turned off by setting the DRCVPA or DRCVBC bits in CSR15. The Physical Address register (CSR12 to CSR14) stores the address that the Am79C972 controller compares to the destination address of the incoming frame for a unicast address match. The Logical Address Filter register (CSR8 to CSR11) serves as a hash filter for multicast address match.

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 01b, which sets the watermark flag at 64 bytes filled.

For test purposes, the Am79C972 controller can be programmed to accept runt packets by setting RPA in CSR124.

Address Matching

The Am79C972 controller supports three types of address matching: unicast, multicast, and broadcast. The normal address matching procedure can be modified by programming three bits in CSR15, the mode register (PROM, DRCVPA, and DRCVBC).

If the first bit received after the SFD (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 1s, 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 Am79C972 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 Physical Address registers (PADR, CSR12 to 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]).

When DRCVPA (CSR15, bit 13) is set to 1, the Am79C972 controller will not accept unicast frames.

If the incoming frame is multicast, the Am79C972 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).

When 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 Am79C972 controller hardware. Broadcast frames are always accepted, except when DRCVBC (CSR15, bit 14) is set and there is no Logical Address match.

None of the address filtering described above applies when the Am79C972 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, CSR15, bit 13).

The Am79C972 controller operates in promiscuous mode when PROM (CSR15, bit 15) is set.

In addition, the Am79C972 controller provides the External Address Detection Interface (EADI) to allow external address filtering. See the section *External Address Detection Interface* for further detail.

The receive descriptor entry RMD1 contains three bits that indicate which method of address matching caused the Am79C972 controller to accept the frame. Note that these indicator bits are only available when the Am79C972 controller is programmed to use 32-bit structures for the descriptor entries (BCR20, bit 7-0, SWSTYLE is set to 2 or 3).

PAM (RMD1, bit 22) is set by the Am79C972 controller when it accepted the received frame due to a match of the frame's destination address with the content of the physical address register.

LAFM (RMD1, bit 21) is set by the Am79C972 controller when it accepted the received frame based on the value in the logical address filter register.

BAM (RMD1, bit 20) is set by the Am79C972 controller when it accepted the received frame because the frame's destination address is of the type 'Broadcast'.

If DRCVBC (CSR15, bit 14) is cleared to 0, only BAM, but not LAFM will be set when a Broadcast frame is received, even if the Logical Address Filter is programmed in such a way that a Broadcast frame would pass the hash filter. If DRCVBC is set to 1 and the Logical Address Filter is programmed in such a way that a Broadcast frame would pass the hash filter, LAFM will be set on the reception of a Broadcast frame.

When the Am79C972 controller operates in promiscuous mode and none of the three match bits is set, it is an indication that the Am79C972 controller only accepted the frame because it was in promiscuous mode.

When the Am79C972 controller is not programmed to be in promiscuous mode, but the EADI interface is enabled, then when none of the three match bits is set, it is an indication that the Am79C972 controller only accepted the frame because it was not rejected by driving the $\overline{\text{EAR}}$ pin LOW within 64 bytes after SFD.

See Table 6 for receive address matches.

Table 6. Receive Address Match

PAM	LAFM	BAM	DRCVBC	Comment
0	0	0	X	Frame accepted due to PROM = 1 or no EADI reject
1	0	0	X	Physical address match
0	1	0	0	Logical address filter match; frame is not of type broadcast
0	1	0	1	Logical address filter match; frame can be of type broadcast
0	0	1	0	Broadcast frame

Automatic Pad Stripping

During reception of an IEEE 802.3 frame, the pad field can be stripped automatically. Setting ASTRP_RCV (CSR4, bit 0) to 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.

Figure 34 shows the byte/bit ordering of the received length field for an IEEE 802.3-compatible frame format.

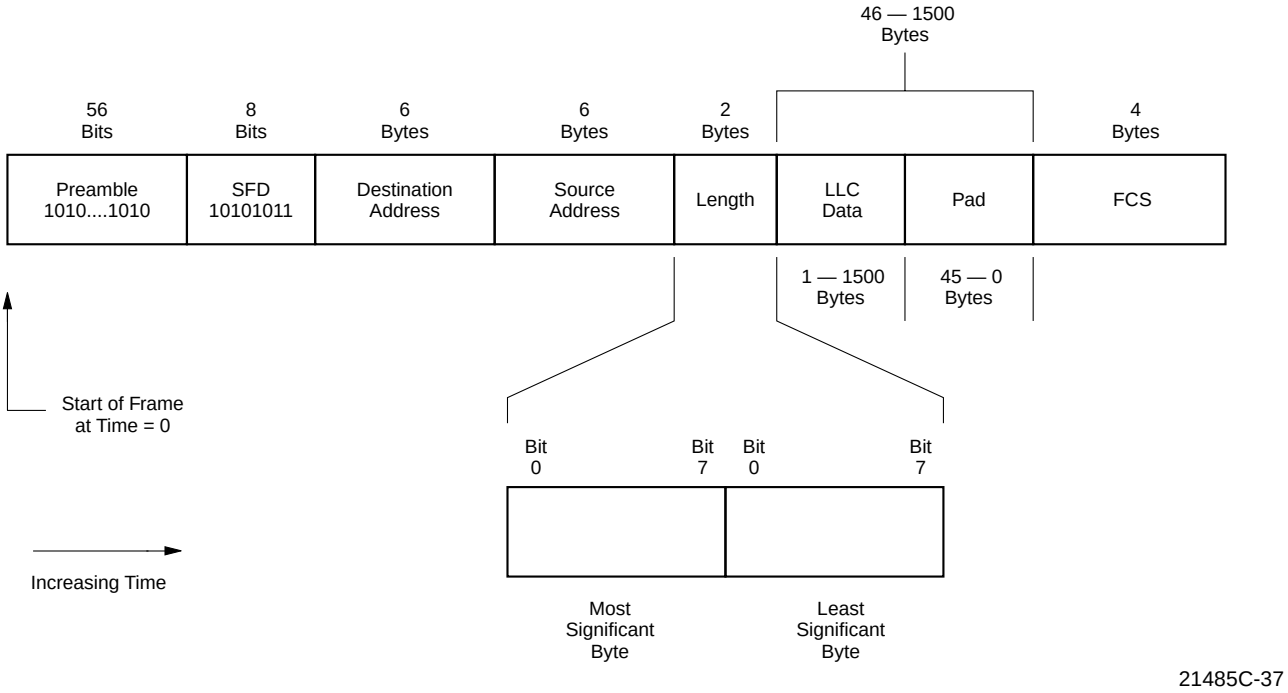


Figure 34. IEEE 802.3 Frame And Length Field Transmission Order

Since any valid Ethernet Type field value will always be greater than a normal IEEE 802.3 Length field (≥46), the Am79C972 controller will not attempt to strip valid Ethernet frames. *Note that for some network protocols, the value passed in the Ethernet Type and/or IEEE 802.3 Length field is not compliant with either standard and may cause problems if pad stripping is enabled*.

Receive FCS Checking

Reception and checking of the received FCS is performed automatically by the Am79C972 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, i.e., those conditions 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 Am79C972 controller are basically collisions within the slot time and automatic runt packet rejection. The Am79C972 controller will ensure that collisions that 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 that is composed of fewer than 64 bytes provided that the Runt Packet Accept (RPA bit in CSR124) feature has not been enabled and the network interface is operating in half-duplex mode, or the full-duplex Runt Packet Accept Disable bit (FDRPAD, BCR9, bit 2) is set. 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 section, *Buffer Management Unit*.

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 basic types of loopback. In internal loopback mode, the transmitted data is looped back to the receiver 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 loopback mode, data can be transmitted to and received from the external network.

Refer to Table 21 for various bit settings required for Loopback modes.

GPSI Loopback Modes

When GPSI is the active network port, there are only two modes of loopback operation: internal and external loopback. Loopback operation is enabled by setting LOOP (CSR15, bit 2) to 1.

When INTL is set to 1, internal loopback is selected. Data coming out of the transmit FIFO is fed directly to the receive FIFO. All GPSI outputs are inactive; inputs are ignored.

External loopback operation is selected by setting INTL to 0. Data is transmitted to the network and is expected to be looped back to the GPSI receive pins outside the chip. Collision detection is active in this mode.

Media Independent Interface Loopback Features

Loopback through the MII can be handled in two ways. The Am79C972 controller supports an internal MII loopback and an external MII loopback. The MII loopback requires that the MII port be manually configured through software using ASEL (BCR 2, bit 1) and PORTSEL (CSR 15, bits 8-7).

The external loopback through the MII requires a two-step operation. The external PHY must be placed into a loopback mode by writing to the MII Control Register (BCR33, BCR34). Then the Am79C972 controller must be placed into an external loopback mode by setting the Loop bits.

The internal loopback through the MII is controlled by MIIILP (BCR32, bit 1). When set to 1, this bit will cause the internal portion of the MII data port to loopback on itself. The MII management port (MDC, MDIO) is unaffected by the MIILP bit. The internal MII interface is mapped in the following way:

- The TXD[3:0] nibble data path is looped back onto the RXD[3:0] nibble data path;
- TX_CLK is looped back as RX_CLK;
- TX_EN is looped back as RX_DV.
- CRS is correctly OR'd with TX_EN and RX_DV and always encompasses the transmit frame.
- TX_ER is not driven by the Am79C972 and therefore not looped back.

During the internal loopback, the TXD, TX_CLK, and TX_EN pins will toggle appropriately with the correct data.

Miscellaneous Loopback Features

All transmit and receive function programming, such as automatic transmit padding and receive pad stripping, operates identically in loopback as in normal operation.

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 C-LANCE (Am79C90) software.

Since the Am79C972 controller has two FCS generators, there are no more restrictions on FCS generation or checking, or on testing multicast address detection as they exist in the half-duplex PCnet family devices and in the C-LANCE. On receive, the Am79C972 controller now provides true FCS status. The descriptor for a frame with an FCS error will have the FCS bit (RMD1, bit 27) set to 1. The FCS generator on the transmit side can still be disabled by setting DXMTFCS (CSR15, bit 3) to 1.

In internal loopback operation, the Am79C972 controller provides a special mode to test the collision logic. When FCOLL (CSR15, bit 4) is set to 1, a collision is forced during every transmission attempt. This will result in a Retry error.

General Purpose Serial Interface

The General Purpose Serial Interface (GPSI) provides a direct interface to the MAC section of the Am79C972 controller. All signals are digital and data is non-encoded. The GPSI allows use of an external Manchester encoder/decoder such as the Am7992B Serial Interface Adapter (SIA). In addition, it allows the Am79C972 controller to be used as a MAC sublayer engine in repeater designs based on the IMR+ device (Am79C981).

GPSI mode is invoked by selecting the interface through the PORTSEL bits of the Mode register (CSR15, bits 8-7).

The GPSI interface uses some of the same pins as the interface to the MII. Simultaneous use of both functions is not possible.

After an H_RESET, all MII pins are internally configured to function as the MII interface. When the GPSI interface is selected by setting PORTSEL (CSR15, bits 8-7) to 10b, the Am79C972 controller will terminate all further accesses to the MII.

GPSI signal functions are described in the pin description section under the GPSI subheading.

Full-Duplex Operation

The Am79C972 controller supports full-duplex operation on both network interfaces. Full-duplex operation

allows simultaneous transmit and receive activity on the TXDAT and RXDAT pins of the GPSI port, and the TXD[3:0] and RXD[3:0] pins of the MII port. Full-duplex operation is enabled by the FDEN bit located in BCR9 for all ports. Full-duplex operation is also enabled through Auto-Negotiation when DANAS (BCR 32, bit 7) is not enabled on the MII port and the ASEL bit is set, and both the external PHY and its link partner are capable of Auto-Negotiation and full-duplex operation.

When operating in full-duplex mode, the following changes to the device operation are made:

Bus Interface/Buffer Management Unit changes:

- The first 64 bytes of every transmit frame are not preserved in the Transmit FIFO during transmission of the first 512 bits as described in the Transmit Exception Conditions section. Instead, when full-duplex mode is active and a frame is being transmitted, the XMTFW bits (CSR80, bits 9-8) always govern when transmit DMA is requested.
- Successful reception of the first 64 bytes of every receive frame is not a requirement for Receive DMA to begin as described in the Receive Exception Conditions section. Instead, receive DMA will be requested as soon as either the RCVFW threshold (CSR80, bits 12-13) is reached or a complete valid receive frame is detected, regardless of length. This Receive FIFO operation is identical to when the RPA bit (CSR124, bit 3) is set during half-duplex mode operation.

The MAC engine changes for full-duplex operation are as follows:

- Changes to the Transmit Deferral mechanism:
  - — Transmission is not deferred while receive is active.
  - — The IPG counter which governs transmit deferral during the IPG between back-to-back transmits is started when transmit activity for the first packet ends, instead of when transmit and carrier activity ends.
- The 4.0 µs carrier sense blinding period after a transmission during which the SQE test normally occurs is disabled.
- The collision indication input to the MAC engine is ignored.

The MII changes for full-duplex operation are as follows:

- The collision detect (COL) pin is disabled.
- The SQE test function is disabled.
- Loss of Carrier (LCAR) reporting is disabled.

Full-Duplex Link Status LED Support

The Am79C972 controller provides bits in each of the LED Status registers (BCR4, BCR5, BCR6, BCR7) to display the Full-Duplex Link Status. If the FDLSE bit (bit 8) is set, a value of 1 will be sent to the associated LED-OUT bit when in Full-Duplex.

Media Independent Interface

The Am79C972 controller fully supports the MII according to the IEEE 802.3 standard. This Reconciliation Sublayer interface allows a variety of PHYs (100BASE-TX, 100BASE-FX, 100BASE-T4, 100BASE-T2, 10BASE-T, etc.) to be attached to the Am79C972 MAC engine without future upgrade problems. The MII interface is a 4-bit (nibble) wide data path interface that runs at 25 MHz for 100-Mbps networks or 2.5 MHz for 10-Mbps networks. The interface consists of two independent data paths, receive (RXD(3:0)) and transmit (TXD(3:0)), control signals for each data path (RX_ER, RX_DV, TX_ER, TX_EN), network status signals (COL, CRS), clocks (RX_CLK, TX_CLK) for each data path, and a two-wire management interface (MDC and MDIO). See Figure 35.

MII Transmit Interface

The MII transmit clock is generated by the external PHY and is sent to the Am79C972 controller on the TX_CLK input pin. The clock can run at 25 MHz or 2.5 MHz, depending on the speed of the network to which the external PHY is attached. The data is a nibble-wide (4 bits) data path, TXD(3:0), from the Am79C972 controller to the external PHY and is synchronous to the rising edge of TX_CLK. The transmit process starts when the Am79C972 controller asserts the TX_EN, which indicates to the external PHY that the data on TXD(3:0) is valid.

Normally, unrecoverable errors are signaled through the MII to the external PHY with the TX_ER output pin. The external PHY will respond to this error by generating a TX coding error on the current transmitted frame. The Am79C972 controller does not use this method of signaling errors on the transmit side. The Am79C972 controller will invert the FCS on the last byte generating an invalid FCS. The TX_ER pin is reserved for future use and is actively driven to 0.

MII Receive Interface

The MII receive clock is also generated by the external PHY and is sent to the Am79C972 controller on the RX_CLK input pin. The clock will be the same frequency as the TX_CLK but will be out of phase and can run at 25 MHz or 2.5 MHz, depending on the speed of the network to which the external PHY is attached.

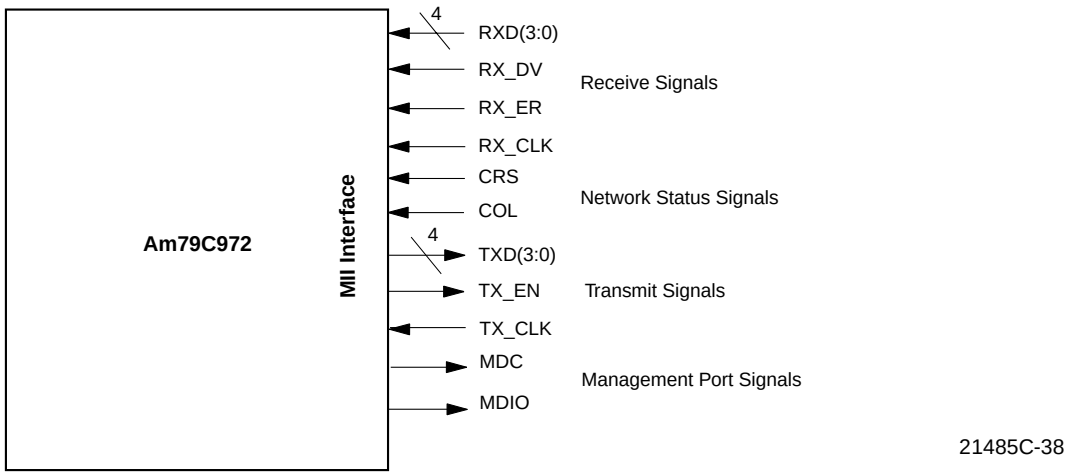


Figure 35. Media Independent Interface

The RX_CLK is a continuous clock during the reception of the frame, but can be stopped for up to two RX_CLK periods at the beginning and the end of frames, so that the external PHY can sync up to the network data traffic necessary to recover the receive clock. During this time, the external PHY may switch to the TX_CLK to maintain a stable clock on the receive interface. The Am79C972 controller will handle this situation with no loss of data. The data is a nibble-wide (4 bits) data path, RXD(3:0), from the external PHY to the Am79C972 controller and is synchronous to the rising edge of RX_CLK.

The receive process starts when RX_DV is asserted. RX_DV will remain asserted until the end of the receive frame. The Am79C972 controller requires CRS (Carrier Sense) to toggle in between frames in order to receive them properly. Errors in the currently received frame are signaled across the MII by the RX_ER pin. RX_ER can be used to signal special conditions *out of band* when RX_DV is not asserted. Two defined out-of-band conditions for this are the 100BASE-TX signaling of *bad* Start of Frame Delimiter and the 100BASE-T4 indication of illegal code group before the receiver has *synched* to the incoming data. The Am79C972 controller will not respond to these conditions. All *out of band*

conditions are currently treated as NULL events. Certain *in band* non-IEEE 802.3u-compliant flow control sequences may cause erratic behavior for the Am79C972 controller. Consult the switch/bridge/router/hub manual to disable the *in-band* flow control sequences if they are being used.

MII Network Status Interface

The MII also provides signals that are consistent and necessary for IEEE 802.3 and IEEE 802.3u operation. These signals are CRS (Carrier Sense) and COL (Collision Sense). Carrier Sense is used to detect non-idle activity on the network. Collision Sense is used to indicate that simultaneous transmission has occurred in a half-duplex network.

MII Management Interface

The MII provides a two-wire management interface so that the Am79C972 controller can control and receive status from external PHY devices.

The Am79C972 controller can support up to 31 external PHYs attached to the MII Management Interface with software support and only one such device without software support.

The Network Port Manager copies the PHYAD after the Am79C972 controller reads the EEPROM and uses it to communicate with the external PHY. The PHY address must be programmed into the EEPROM prior to starting the Am79C972 controller. This is necessary so that the internal management controller can work autonomously from the software driver and can always know where to access the external PHY. The Am79C972 controller is unique by offering direct hardware support of the external PHY device without software support. The PHY address of 1Fh is reserved and should not be used. To access the 31 external PHYs, the software driver must have knowledge of the external PHY's address when multiple PHYs are present before attempting to address it.

The MII Management Interface uses the MII Control, Address, and Data registers (BCR32, 33, 34) to control and communicate to the external PHYs. The Am79C972 controller generates MII management frames to the external PHY through the MDIO pin synchronous to the rising edge of the Management Data Clock (MDC) based on a combination of writes and reads to these registers.

MII Management Frames

MII management frames are automatically generated by the Am79C972 controller and conform to the MII clause in the IEEE 802.3u standard.

The start of the frame is a preamble of 32 ones and guarantees that all of the external PHYs are synchronized on the same interface. (See Figure 36.) Loss of synchronization is possible due to the *hot-plugging* capability of the exposed MII.

The IEEE 802.3 specification allows you to drop the preamble, if after reading the MII Status Register from the external PHY you can determine that the external PHY will support Preamble Suppression (BCR34, bit 6). After having a valid MII Status Register read, the Am79C972 controller will then drop the creation of the preamble stream until a reset occurs, receives a read error, or the external PHY is disconnected.

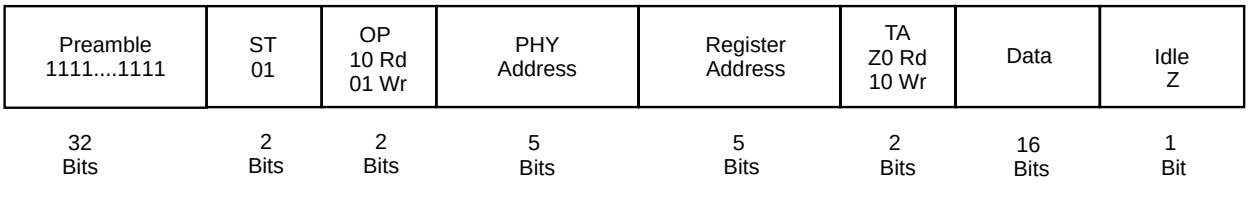


Figure 36. Frame Format at the MII Interface Connection

This is followed by a start field (ST) and an operation field (OP). The operation field (OP) indicates whether the Am79C972 controller is initiating a read or write operation. This is followed by the external PHY address (PHYAD) and the register address (REGAD) programmed in BCR33. The PHY address of 1Fh is reserved and should not be used. The external PHY may have a larger address space starting at 10h - 1Fh. This is the address range set aside by the IEEE as vendor usable address space and will vary from vendor to vendor. This field is followed by a bus turnaround field. During a read operation, the bus turnaround field is used to determine if the external PHY is responding correctly to the read request or not. The Am79C972 controller will tri-state the MDIO for both MDC cycles.

During the second cycle, if the external PHY is synchronized to the Am79C972 controller, the external PHY will drive a 0. If the external PHY does not drive a 0, the Am79C972 controller will signal a MREINT (CSR7, bit 9) interrupt, if MREINTE (CSR7, bit 8) is set

to a 1, indicating the Am79C972 controller had an MII management frame read error and that the data in BCR34 is not valid. The data field to/from the external PHY is read or written into the BCR34 register. The last field is an IDLE field that is necessary to give ample time for drivers to turn off before the next access. The Am79C972 controller will drive the MDC to 0 and tri-state the MDIO anytime the MII Management Port is not active.

To help to speed up the reading and writing of the MII management frames to the external PHY, the MDC can be sped up to 10 MHz by setting the FMDC bits in BCR32. The IEEE 802.3 specification requires use of the 2.5-MHz clock rate, but 5 MHz and 10 MHz are available for the user. The intended applications are that the 10-MHz clock rate can be used for a single external PHY on an adapter card or motherboard. The 5-MHz clock rate can be used for an exposed MII with one external PHY attached. The 2.5-MHz clock rate is intended to be used when multiple external PHYs are connected to the MII Management Port or if compliance to the IEEE 802.3u standard is required.

Auto-Poll External PHY Status Polling

As defined in the IEEE 802.3 standard, the external PHY attached to the Am79C972 controller's MII has no way of communicating important timely status information back to Am79C972 controller. The Am79C972 controller has no way of knowing that an external PHY has undergone a change in status without polling the MII status register. To prevent problems from occurring with inadequate host or software polling, the Am79C972 controller will Auto-Poll when APEP (BCR32, bit 11) is set to 1 to insure that the most current information is available. See *Appendix C, MII Management Registers,* for the bit descriptions of the MII Status Register. The contents of the latest read from the external PHY will be stored in a shadow register in the Auto-Poll block. The first read of the MII Status Register will just be stored, but subsequent reads will be compared to the contents already stored in the shadow register. If there has been a change in the contents of the MII Status Register, a MAPINT (CSR7, bit 7) interrupt will be generated on $\overline{\text{INTA}}$ if the MAPINTE (CSR7, bit 6) is set to 1. The Auto-Poll features can be disabled if software driver polling is required.

The Auto-Poll's frequency of generating MII management frames can be adjusted by setting of the APDW bits (BCR32, bits 10-8). The delay can be adjusted from 0 MDC periods to 2048 MDC periods. Auto-Poll by default will only read the MII Status register in the external PHY.

Network Port Manager

The Am79C972 controller is unique in that it does not require software intervention to control and configure an external PHY attached to the MII. This was done to ensure backwards compatibility with existing software drivers. To the current software drivers, the Am79C972 controller will look and act like the PCnet-PCI II and will interoperate with existing PCnet drivers from revision 2.5 upward. The heart of this system is the Network Port Manager.

If the external PHY is present and is active, the Network Port Manager will request status from the external PHY by generating MII management frames. These frames will be sent roughly every 900 ms. These frames are necessary so that the Network Port Manager can monitor the current active link and can select a different network port if the current link goes down.

Auto-Negotiation

Through the external PHY, the following capabilities are possible: 100BASE-T4, 100BASE-TX Full-/Half-Duplex, and 10BASE-T Full-/Half-Duplex. The capabilities are then sent to a link partner that will also send its capabilities. Both sides look to see what is possible and then they will connect at the greatest possible speed and capability as defined in the IEEE 802.3u standard and according to Table 7.

By default, the link partner must be at least 10BASE-T half-duplex capable. The Am79C972 controller can automatically negotiate with the network and yield the highest performance possible without software support. See the section on *Network Port Manager* for more details.

Table 7. Auto-Negotiation Capabilities

Network Speed	Physical Network Type
200 Mbps	100BASE-X, Full Duplex
100 Mbps	100BASE-T4, Half Duplex
100 Mbps	100BASE-X, Half Duplex
20 Mbps	10BASE-T, Full Duplex
10 Mbps	10BASE-T, Half Duplex

Auto-Negotiation goes further by providing a message-based communication scheme called, *Next Pages*, before connecting to the Link Partner. *This feature is not supported in Am79C972 unless the DANAS (BCR32, bit 10) is selected and the software driver is capable of controlling the external PHY.* A complete bit description of the MII and Auto-Negotiation registers can be found in Appendix C.

Automatic Network Port Selection

If ASEL (BCR2, bit 0) is set to 1 and DANAS (BCR 32, bit 7) is set to 0, then the Network Port Manager will start to configure the external PHY if it detects the external PHY on the MII Interface.

Automatic Network Selection: Exceptions

If ASEL (BCR2, bit 0) is set to 0 or DANAS (BCR 32, bit 7) is set to 1, then the Network Port Manager will discontinue actively trying to establish the connections. It is assumed that the software driver is attempting to configure the network port and the Am79C972 controller will always defer to the software driver. When The ASEL is set to 0, the software driver should then configure the ports with PORTSEL (CSR15, bits 7-8). The GPSI does not participate in the automatic selection process and should be manually configured with the PORTSEL bits.

Note: *It is highly recommended that ASEL and PORTSEL be used when trying to manually configure a specific network port.*

In order to manually configure the External PHY, the ***recommended procedure*** is to force the PHY configurations when Auto-Negotiation is *not* enabled. Set the DANAS bit (BCR32, bit 7) to turn off the Network Port Manager. Then write again to BCR32 with the DANAS and XPHANE (BCR32, bit 5) bits cleared, together with the XPHYFD (BCR32, bit 4) and XPHYSP (BCR32, bit 3) bits set to the desired configuration. The Network Port Manager will send a few frames to validate the configuration.

CAUTION: *The Network Port Manager utilizes the PHYADD (BCR33, bits 9-5) to communicate with the external PHY during the automatic port selection process. The PHYADD is copied into a shadow register after the Am79C972 controller has read the configuration information from the EEPROM. Extreme care must be exercised by the host software not to access BCR33 during this time. A read of PVALID (BCR19, bit 15) before accessing BCR33 will guarantee that the PHYADD has been shadowed.*

Am79C972's Automatic Network Port selection mechanism falls within the following general categories:

- External PHY *Not* Auto-Negotiable
- External PHY Auto-Negotiable

Automatic Network Selection: External PHY *Not* Auto-Negotiable

This case occurs when the MIIPD (BCR32, bit 14) bit is 1. This indicates that there is an external PHY attached to Am79C972 controller's MII. If more than one external PHY is attached to the MII Management Interface, then the DANAS (BCR32, bit 7) bit must be set to 1 and then all configuration control should revert to software. The Am79C972 controller will read the register of the external PHY to determine its status and network capabilities. See *Appendix C, MII Management Registers,* for the bit descriptions of the MII Status register. If the external PHY is not Auto-Negotiation capable and/or the XPHYANE (BCR32, bit 5) bit is set to 0, then the Network Port Manager will match up the external PHY capabilities with the XPHYFD (BCR 32, bit 4) and the XPHYSP (BCR32, bit 3) bits programmed from the EEPROM. The Am79C972 controller will then program the external PHY with those values. A new read of the external PHYs MII Status register will be made to see if the link is up. If the link does not come up as programmed after a specific time, the Am79C972 controller will fail the external PHY link. The Network Port Manager will periodically query the external PHY for active links.

Automatic Network Selection: External PHY Auto-Negotiable

This case occurs when the MIIPD (BCR32, bit 14) bit is 1. This indicates that there is an external PHY attached to Am79C972 controller's MII. If more than one external PHY is attached to the MII Management Interface, then the DANAS (BCR32, bit 7) bit must be set to 1 and then all configuration control should revert to software. The Am79C972 controller will read the MII Status register of the external PHY to determine its status and network capabilities. See Appendix C for the bit descriptions of the MII Status register. If the external PHY is Auto-Negotiation capable and/or the XPHYANE (BCR32, bit 5) bit is set to 1, then the Am79C972 controller will start the external PHY's Auto-Negotiation process. The Am79C972 controller will write to the external PHY's Advertisement register with the following conditions set: turn off the Next Pages support, set the Technology Ability Field (See Appendix C for the Auto-Negotiation register bit descriptions) from the external PHY MII Status register read, and set the Type Selector field to the IEEE 802.3 standard. The Am79C972 controller will then write to the external PHY's MII Control register instructing the external PHY to negotiate the link. The Am79C972 controller will poll the external PHY's MII Status register until the Auto-Negotiation Complete bit is set to 1and the Link Status bit is set to 1. The Am79C972 controller will then wait a specific time and then again read the external PHY's MII Status register. If the Am79C972 controller sees that the external PHY's link is down, it will try to bring up the external PHY's link manually as described above. A new read of the external PHY's MII Status register will be made to see if the link is up. If the link does not come up as programmed after a specific time, the Am79C972 controller will fail the external PHY link and start the process again.

Automatic Network Selection: Force External Reset

If the XPHYRST bit (BCR32, bit 6) is set to 1, then the flow changes slightly. The Am79C972 controller will write to the external PHY's MII Control register with the RESET bit set to 1 (See *Appendix C, MII Management Registers,* for the MII register bit descriptions). This will force a complete reset of the external PHY. The Am79C972 controller after a specific time will poll the external PHY's MII Control register to see if the RESET

bit is 0. After the RESET bit is cleared, then the normal flow continues.

External Address Detection Interface

The EADI is provided to allow external address filtering and to provide a Receive Frame Tag word for proprietary routing information. It is selected by setting the EADISEL bit in BCR2 to 1. This feature is typically utilized by terminal servers, bridges and/or router products. The EADI interface can be used in conjunction with external logic to capture the packet destination address from the serial bit stream as it arrives at the Am79C972 controller, to compare the captured address with a table of stored addresses or identifiers, and then to determine whether or not the Am79C972 controller should accept the packet.

External Address Detection Interface: GPSI Port

The EADI interface outputs are delivered directly from the NRZ decoded data and clock recovered by the external PHY. This allows the external address detection to be performed in parallel with frame reception and address comparison in the MAC Station Address Detection (SAD) block of the Am79C972 controller.

SRDCLK is provided to allow clocking of the receive bit stream into the external address detection logic. Once a received frame commences and data and clock are available, the EADI logic will monitor the alternating (“1,0”) preamble pattern until the two 1s of the Start Frame Delimiter (SFD, 10101011 bit pattern) are detected, at which point the SFBD output will be driven HIGH.

The SFBD signal will initially be LOW. The assertion of SFBD is a signal to the external address detection logic that the SFD has been detected and that subsequent SRDCLK cycles will deliver packet data to the external logic. Therefore, when SFBD is asserted, the external address matching logic should begin de-serialization of the SRD data and send the resulting destination address to a Content Addressable Memory (CAM) or other address detection device. In order to reduce the amount of logic external to the Am79C972 controller for multiple address decoding systems, the SFBD signal will toggle at each new byte boundary within the packet, subsequent to the SFD. This eliminates the need for externally supplying byte framing logic.

SRD is the decoded NRZ data from the network. This signal can be used for external address detection.

The $\overline{\text{EAR}}$ pin should be driven LOW by the external address comparison logic to reject a frame.

If an address match is detected by comparison with either the Physical Address or Logical Address Filter registers contained within the Am79C972 controller or the frame is of the type 'Broadcast', then the frame will be accepted regardless of the condition of $\overline{\text{EAR}}$. When the EADISEL bit of BCR2 is set to 1 and the Am79C972 controller is programmed to promiscuous mode (PROM bit of the Mode Register is set to 1), then all incoming frames will be accepted, regardless of any activity on the $\overline{\text{EAR}}$ pin.

Internal address match is disabled when PROM (CSR15, bit 15) is cleared to 0, DRCVBC (CSR15, bit 14) and DRCVPA (CSR15, bit 13) are set to 1, and the Logical Address Filter registers (CSR8 to CSR11) are programmed to all zeros.

When the EADISEL bit of BCR2 is set to 1 and internal address match is disabled, then all incoming frames will be accepted by the Am79C972 controller, unless the $\overline{\text{EAR}}$ pin becomes active during the first 64 bytes of the frame (excluding preamble and SFD). This allows external address lookup logic approximately 58 byte times after the last destination address bit is available to generate the $\overline{\text{EAR}}$ signal, assuming that the Am79C972 controller is not configured to accept runt packets. The EADI logic only samples $\overline{\text{EAR}}$ from 2 bit times after SFD until 512 bit times (64 bytes) after SFD. The frame will be accepted if $\overline{\text{EAR}}$ has not been asserted during this window. In order for the $\overline{\text{EAR}}$ pin to be functional in full-duplex mode, FDRPAD bit (BCR9, bit 2) needs to be set. If Runt Packet Accept (CSR124, bit 3) is enabled, then the $\overline{\text{EAR}}$ signal must be generated prior to the 8 bytes received, if frame rejection is to be guaranteed. Runt packet sizes could be as short as 12 byte times (assuming 6 bytes for source address, 2 bytes for length, no data, 4 bytes for FCS) after the last bit of the destination address is available. $\overline{\text{EAR}}$ must have a pulse width of at least 110 ns.

The EADI outputs continue to provide data throughout the reception of a frame. This allows the external logic to capture frame header information to determine protocol type, internetworking information, and other useful data.

The EADI interface will operate as long as the STRT bit in CSR0 is set, even if the receiver and/or transmitter are disabled by software (DTX and DRX bits in CSR15 are set). This configuration is useful as a semi-power-down mode in that the Am79C972 controller will not perform any power-consuming DMA operations. However, external circuitry can still respond to *control* frames on the network to facilitate remote node control. Table 8 summarizes the operation of the EADI interface.

Table 8. EADI Operations

PROM	EAR	Required Timing	Received Frames
1	X	No timing requirements	All received frames
0	1	No timing requirements	All received frames
0	0	Low for two bit times plus 10 ns	Frame rejected if in address match mode

External Address Detection Interface: External PHY

When using the MII, the EADI interface changes to reflect the changes on that interface. Except for the notations below the interface conforms to the previous functionality. The data arrives in nibbles and can be at a rate of 25 MHz or 2.5 MHz.

The MII provides all necessary data and clock signals needed for the EADI interface. Consequently, SRDCLK and SRD are not used and are driven to 0. Data for the EADI is the RXD(3:0) receive data provided to the MII. Instead of deserializing the network data, the user will receive the data as 4 bit nibbles. RX_CLK is provided to allow clocking of the RXD(3:0) receive nibble stream into the external address detection logic. The RXD(3:0) data is synchronous to the rising edge of the RX_CLK.

The assertion of SFBD is a signal to the external address detection logic that the SFD has been detected and that the first valid data nibble is on the RXD(3:0) data bus. The SFBD signal is delayed one RX_CLK cycle from the above definition and actually signals the start of valid data. In order to reduce the amount of logic external to the Am79C972 controller for multiple address decoding systems, the SFBD signal will go HIGH at each new byte boundary within the packet, subsequent to the SFD. This eliminates the need for externally supplying byte framing logic.

The $\overline{\text{EAR}}$ pin function is the same and should be driven LOW by the external address comparison logic to reject a frame. See the *External Address Detection Interface: GPSI Port* section for more details.

External Address Detection Interface: Receive Frame Tagging

The Am79C972 controller supports receive frame tagging in both GPSI or MII mode. The method remains constant, but the chip interface pins will change between the MII and the GPSI modes. The receive frame tagging implementation will be a two- and three-wire chip interface, respectively, added to the existing EADI.

The Am79C972 controller supports up to 15 bits of receive frame tagging per frame in the receive frame status (RFRTAG). The RFRTAG bits are in the receive frame status field in RMD2 (bits 30-16) in 32-bit software mode. The receive frame tagging is not supported in the 16-bit software mode. The RFRTAG field are all zeros when either the EADISEL (BCR2, bit3) or the RXFRTAG (CSR7, bit 14) are set to 0. When EADISEL (BCR2, bit 3) and RXFRTAG (CSR7, bit 14) are set to 1, then the RFRTAG reflects the tag word shifted in during that receive frame.

In the MII mode, the two-wire interface will use the MIIRXFRTGD and MIIRXFRTGE pins from the EADI interface. These pins will provide the data input and data input enable for the receive frame tagging, respectively. These pins are normally not used during the MII operation.

In the GPSI mode, the three-wire interface will use the RXFRTGD, SRDCLK, and the RXFRTGE pins from the EADI and MII. These pins will provide the data input, data input clock, and the data input for the receive frame tagging enable, respectively.

The receive frame tag register is a shift register that shifts data in MSB first, so that less than the 15 bits allocated may be utilized by the user. The upper bits not utilized will return zeros. The receive frame tag register is set to 0 in between reception of frames. After receiving SFBD indication on the EADI, the user can start shifting data into the receive tag register until one network clock period before the Am79C972 controller receives the end of the current receive frame.

In the MII mode, the user must see the RX_CLK to drive the synchronous receive frame tag data interface. After receiving the SFBD indication, sampled by the rising edge of the RX_CLK, the user will drive the data input and the data input enable synchronous with the rising edge of the RX_CLK. The user has until one network clock period before the deassertion of the RX_DV to input the data into the receive frame tag register. At the deassertion of the RX_DV, the receive frame tag register will no longer accept data from the two-wire interface. If the user is still driving the data input enable pin, erroneous or corrupted data may reside in the receive frame tag register. See Figure 37.

In the GPSI mode, the user must use the recovered receive data clock driven on the SRDCLK pin to drive the synchronous receive frame tag data interface. After receiving the SFBD indication, sampled by the rising edge of the recovered receive data clock, the user will drive the data input and the data input enable synchronous with the rising edge of the recovered receive data clock. The user has until one network clock period before the deassertion of the data from the network to input the data into the receive frame tag register. At the completion of received network data, the receive frame tag register will no longer accept data from the two-wire interface. If the user is still driving the data input enable pin, erroneous or corrupted data may reside in the receive frame tag register. See Figure 38.

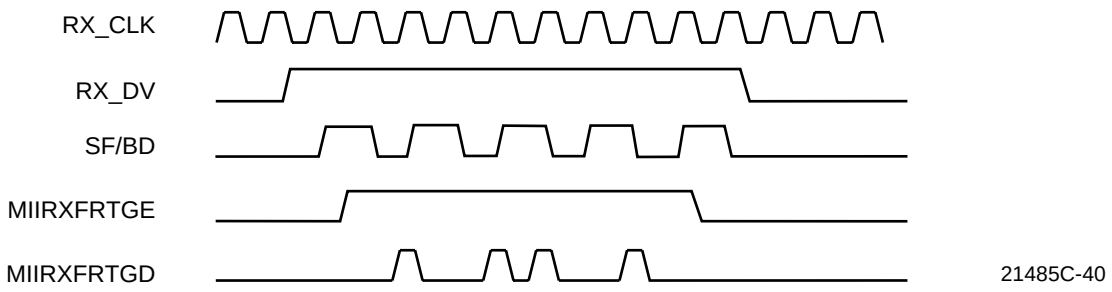
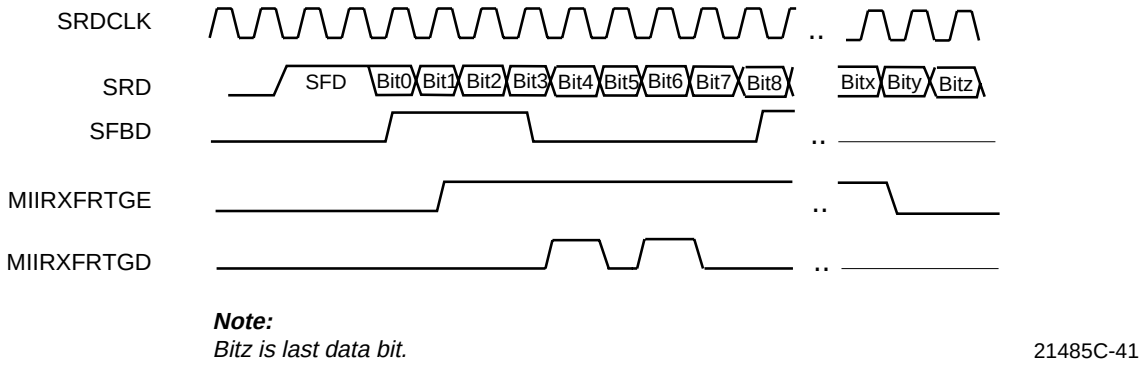


Figure 37. MII Receive Frame Tagging



Note:
Bitz is last data bit.

Figure 38. GPSI Mode Frame Tagging

Expansion Bus Interface

The Am79C972 controller contains an Expansion Bus Interface that supports Flash and EPROM devices as boot devices, as well as provides read/write access to Flash or EPROM.

The signal AS_$\overline{\text{EBOE}}$ is provided to strobe the upper 8 bits of the address into an external ‘374 (D flip-flop) address latch. AS_$\overline{\text{EBOE}}$ is asserted LOW during EPROM/Flash read operations to control the $\overline{\text{OE}}$ input of the EPROM/Flash.

The Expansion Bus Address is split into two different buses, EBUA_EBA[7:0] and EBDA[15:8]. The EBUA_EBA[7:0] provides the least and the most significant address byte. When accessing EPROM/Flash, the EBUA_EBA[7:0] is strobed into an external ‘374 (D flip-flop) address latch. This constitutes the most significant portion of the Expansion Bus Address. For EPROM/Flash accesses, EBUA_EBA[7:0] constitutes the remaining least significant address byte. For byte oriented EPROM/Flash accesses, EBDA[15:8] constitutes the upper or middle address byte. EBADDRU (BCR29, bits 3-0) should be set to 0 when not used, since EBADDRU constitutes the EBUA portion of the EBUA_EBA address byte and is strobed into the external ’374 address latch.

The signal $\overline{\text{EROMCS}}$ is connected to the $\overline{\text{CS}}$/$\overline{\text{CE}}$ input of the EPROM/Flash. The signal $\overline{\text{EBWE}}$ is connected to the $\overline{\text{WE}}$ of the Flash device.

The Expansion Data Bus is configured for 8-bit byte access during EPROM/Flash accesses. During EPROM/Flash accesses, EBD[7:0] provides the data byte. See Figure 39, Figure 40, and Figure 41.

Expansion ROM - Boot Device Access

The Am79C972 controller supports EPROM or Flash as an Expansion ROM boot device. Both are configured using the same methods and operate the same. See the previous section on Expansion ROM transfers to get the PCI timing and functional description of the transfer method. The Am79C972 controller is functionally equivalent to the PCnet-PCI II controller with Expansion ROM. See Figure 40 and Figure 41.

The Am79C972 controller will always read four bytes for every host Expansion ROM read access. The interface to the Expansion Bus runs synchronous to the PCI bus interface clock. The Am79C972 controller will start the read operation to the Expansion ROM by driving the upper 8 bits of the Expansion ROM address on EBUA_EBA[7:0]. One-half clock later, AS_$\overline{\text{EBOE}}$ goes high to allow registering of the upper address bits externally. The upper portion of the Expansion ROM address will be the same for all four byte read cycles.

AS_EBOE is driven high for one-half clock, EBUA_EBA[7:0] are driven with the upper 8 bits of the Expansion ROM address for one more clock cycle after AS_EBOE goes low. Next, the Am79C972 controller starts driving the lower 8 bits of the Expansion ROM address on EBUA_EBA[7:0].

The time that the Am79C972 controller waits for data to be valid is programmable. ROMTMG (BCR18, bits 15-12) defines the time from when the Am79C972 controller drives EBUA_EBA[7:0] with the lower 8 bits of the Expansion ROM address to when the Am79C972 controller latches in the data on the EBD[7:0] inputs. The register value specifies the time in number of clock cycles. When ROMTMG is set to nine (the default value), EBD[7:0] is sampled with the next rising edge of CLK ten clock cycles after EBUA_EBA[7:0] was driven with a new address value. The clock edge that is used to sample the data is also the clock edge that generates the next Expansion ROM address. All four bytes of Expansion ROM data are stored in holding registers. One clock cycle after the last data byte is available, the Am79C972 controller asserts TRDY.

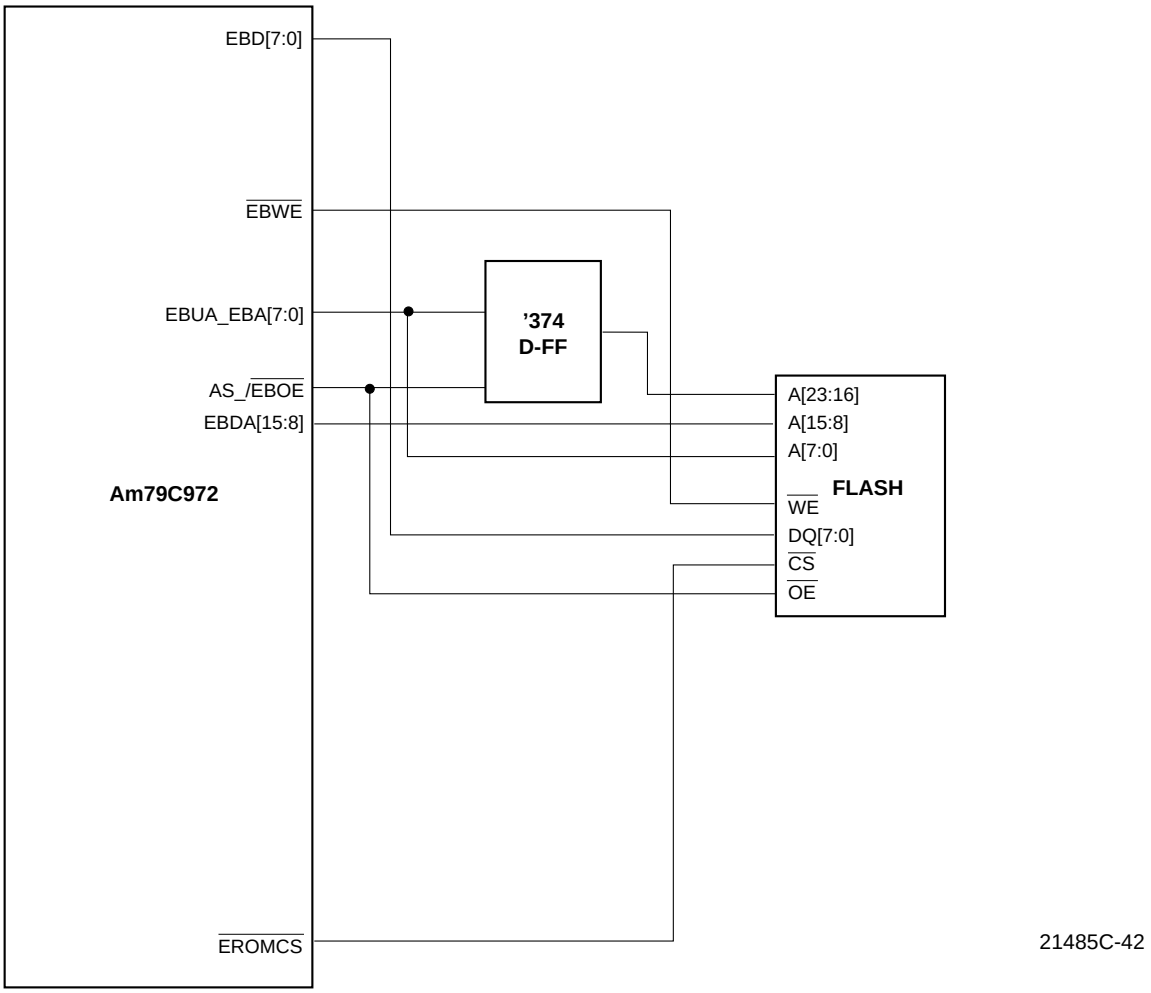


Figure 39. Flash Configuration for the Expansion Bus

The access time for the Expansion ROM or the EBDATA (BCR30) device (tACC) during read operations can be calculated by subtracting the clock to output delay for the EBUA_EBA[7:0] outputs (tv_A_D) and by subtracting the input to clock setup time for the EBD[7:0] inputs (ts_D) from the time defined by ROMTMG:

tACC = ROMTMG * CLK period *CLK_FAC - (tv_A_D) - (ts_D)

The access time for the Expansion ROM or for the EBDATA (BCR30) device (tACC) during write operations can be calculated by subtracting the clock to output delay for the EBUA EBA[7:0] outputs (tv_A_D) and by

adding the input to clock setup time for Flash/EPRO inputs (ts_D) from the time defined by ROMTMG:

tACC = ROMTMG * CLK period * CLK_FAC - (tv_A_D) - (ts_D)

The timing diagram in Figure 42 assumes the default programming of ROMTMG (1001b = 9 CLK). After reading the first byte, the Am79C972 controller reads in three more bytes by incrementing the lower portion of the ROM address. After the last byte is strobed in, TRDY will be asserted on clock 50. When the host tries to perform a burst read of the Expansion ROM, the Am79C972 controller will disconnect the access at the second data phase.

The host must program the Expansion ROM Base Address register in the PCI configuration space before the first access to the Expansion ROM. The Am79C972 controller will not react to any access to the Expansion ROM until both MEMEN (PCI Command register, bit 1) and ROMEN (PCI Expansion ROM Base Address register, bit 0) are set to 1. After the Expansion ROM is enabled, the Am79C972 controller will claim all memory read accesses with an address between ROMBASE and ROMBASE + 1M - 4 (ROMBASE, PCI Expansion ROM Base Address register, bits 31-20). The address output to the Expansion ROM is the offset from the address on the PCI bus to ROMBASE. The Am79C972 controller aliases all accesses to the Expansion ROM of the command types *Memory Read Multiple* and *Memory Read Line* to the basic Memory Read command.

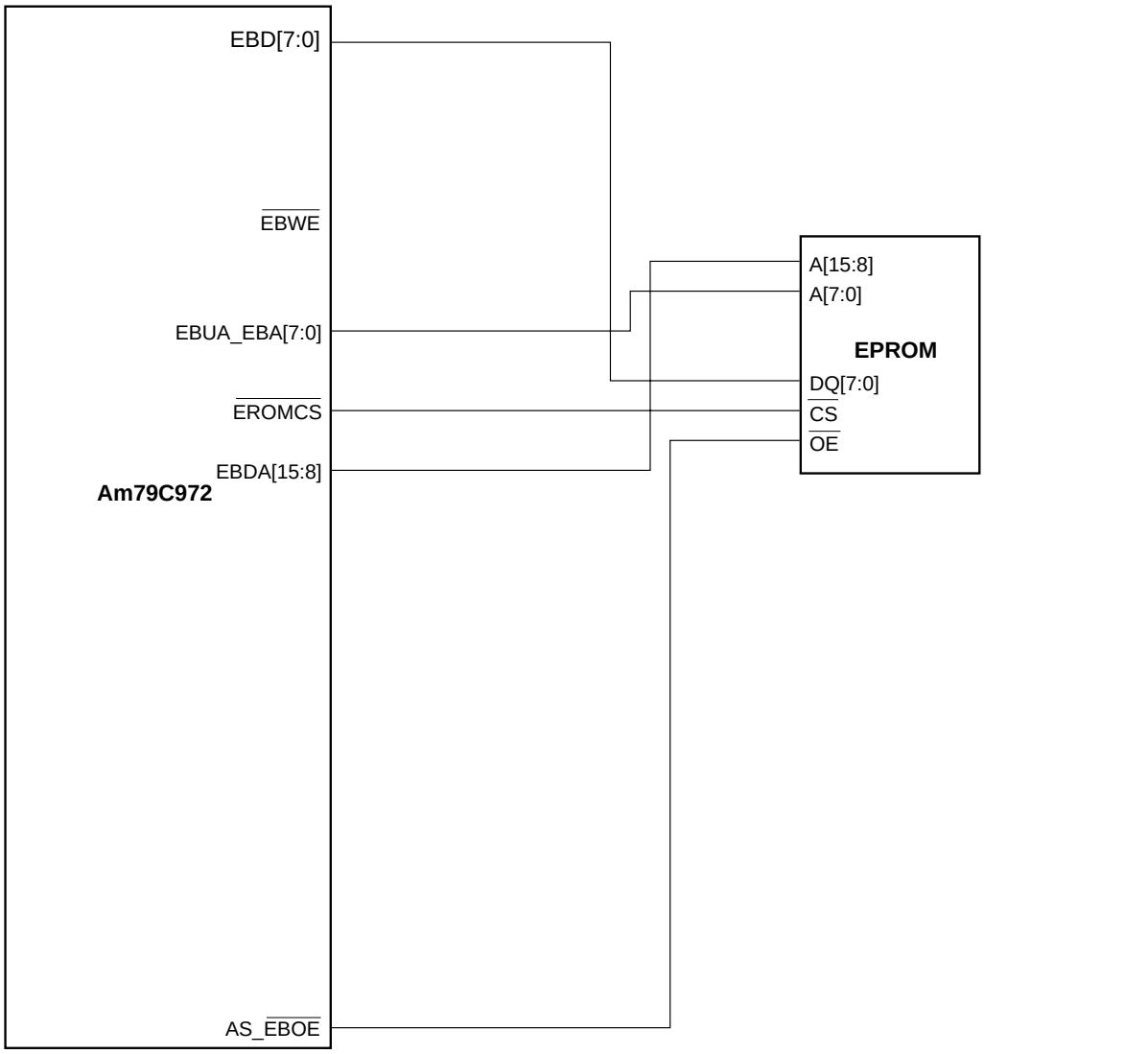


Figure 40. EPROM Only Configuration for the Expansion Bus (64K EPROM)

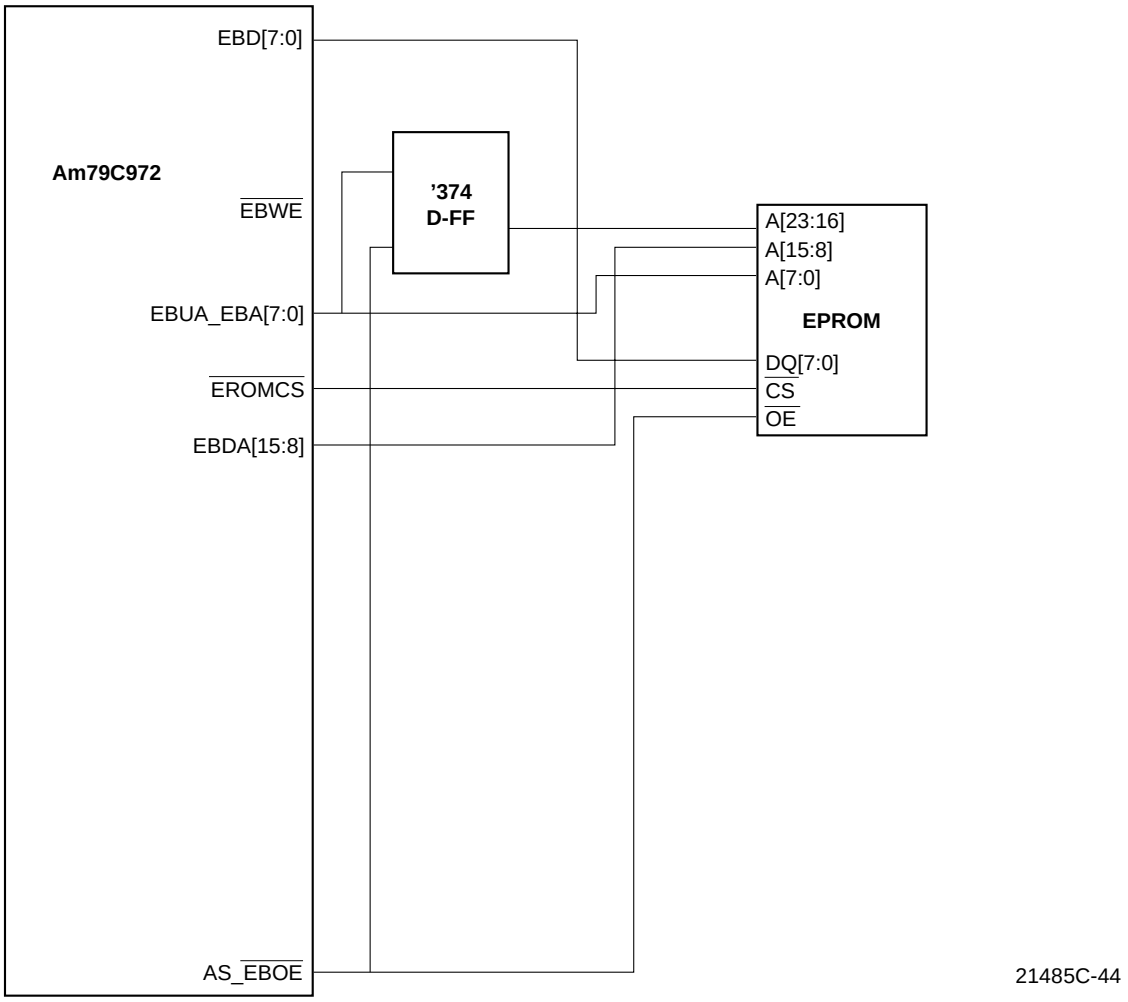


Figure 41. EPROM Only Configuration for the Expansion Bus (>64K EPROM)

Since setting MEMEN also enables memory mapped access to the I/O resources, attention must be given to the PCI Memory Mapped I/O Base Address register, before enabling access to the Expansion ROM. The host must set the PCI Memory Mapped I/O Base Address register to a value that prevents the Am79C972 controller from claiming any memory cycles not intended for it.

During the boot procedure, the system will try to find an Expansion ROM. A PCI system assumes that an Expansion ROM is present when it reads the ROM signature 55h (byte 0) and AAh (byte 1).

Direct Flash Access

Am79C972 controller supports Flash as an Expansion ROM device, as well as providing a read/write data path to the Flash. The Am79C972 controller will support up to 1 Mbyte of Flash on the Expansion Bus. The Flash is accessed by a read or write to the Expansion Bus Data port (BCR30). The user must load the upper address EPADDRU (BCR 29, bits 3-0) and then set the FLASH (BCR29, bit 15) bit to a 1. The Flash read/write utilizes the PCI clock instead of the EBCLK during all accesses. EPADDRU is not needed if the Flash size is 64K or less, but still must be programmed. The user will then load the lower 16 bits of address, EPADDRL (BCR 28, bits 15-0).

Flash/EPROM Read

A read to the Expansion Bus Data Port (BCR30) will start a read cycle on the Expansion Bus Interface. The Am79C972 controller will drive EBUA_EBA[7:0] with the most significant address byte at the same time the Am79C972 controller will drive AS_$\overline{\text{EBOE}}$ high to strobe the address in the external '374 (D flip-flop). On the next clock, the Am79C972 controller will drive EBDA[15:8] and EBUA_EBA[7:0] with the middle and least significant address bytes.

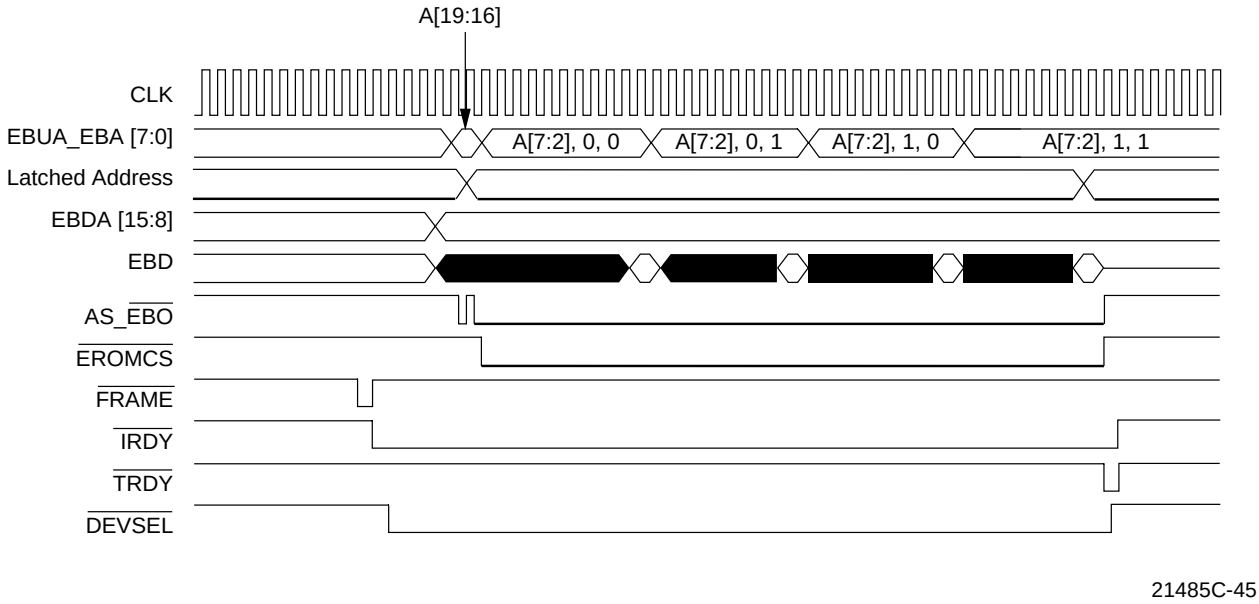


Figure 42. Expansion ROM Bus Read Sequence

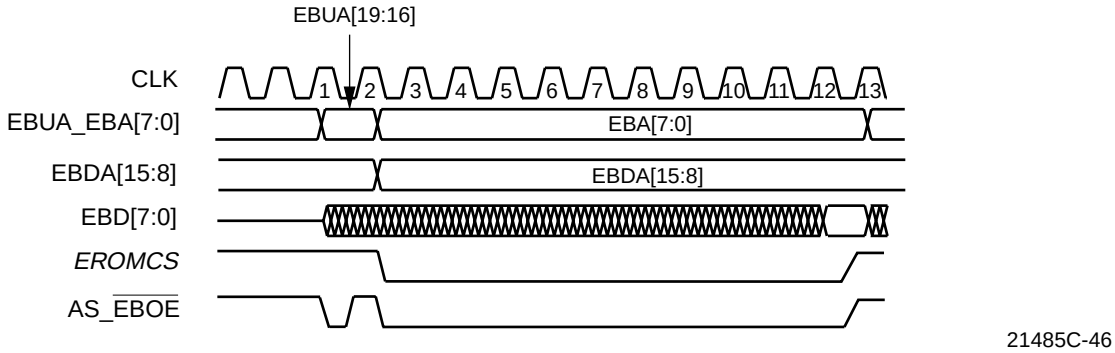


Figure 43. Flash Read from Expansion Bus Data Port

The EROMCS is driven low for the value ROMTMG + 1. Figure 43 assumes that ROMTMG is set to nine. EBD[7:0] is sampled with the next rising edge of CLK ten clock cycles after EBUA_EBA[7:0] was driven with a new address value. This PCI slave access to the Flash/EPROM will result in a retry for the very first access. Subsequent accesses may give a retry or not, depending on whether or not the data is present and valid. The access time is dependent on the ROMTMG bits (BCR18, bits 15-12) and the Flash/EPROM. This access mechanism differs from the Expansion ROM access mechanism since only one byte is read in this manner, instead of the 4 bytes in an Expansion ROM access. The PCI bus will not be held during accesses through the Expansion Bus Data Port. If the LAAINC (BCR29, bit 15) is set, the EBADDRL address will be incremented and a continuous series of reads from the Expansion Data Port (EBDATA, BCR30) is possible. The address incrementor will roll over without warning and without incrementing the upper address EBADDRU.

The Flash write is almost the same procedure as the read access, except that the Am79C972 controller will not drive AS_EBOE low. The EROMCS and EBWE are driven low for the value ROMTMG again. The write to the FLASH port is a posted write and will not result in a retry to the PCI unless the host tries to write a new value before the previous write is complete, then the host will experience a retry. See Figure 44.

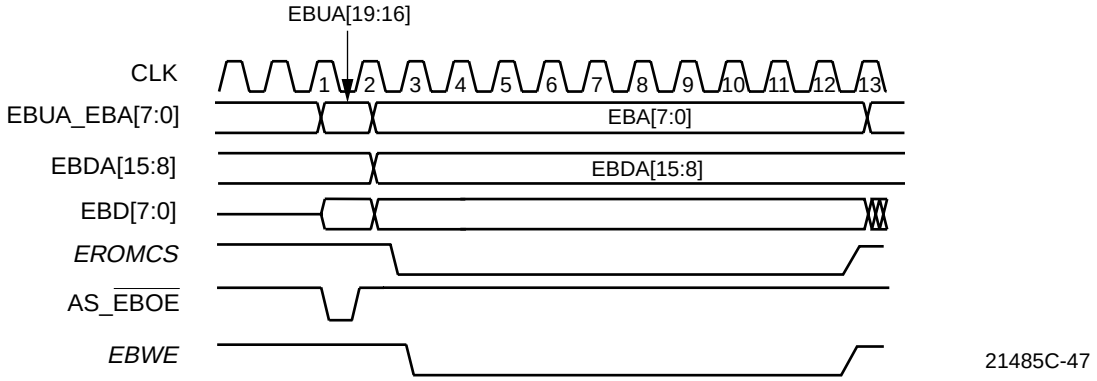


Figure 44. Flash Write from Expansion Bus Data Port

AMD Flash Programming

AMD's Flash products are programmed on a byte-by-byte basis. Programming is a four bus cycle operation. There are two "unlock" write cycles. These are followed by the program set-up command and data write cycles. Addresses are latched on the falling edge of $\overline{\text{EBWE}}$ and the data is latched on the rising edge of $\overline{\text{EBWE}}$. The rising edge of $\overline{\text{EBWE}}$ begins programming.

Upon executing the AMD Flash Embedded Program Algorithm command sequence, the Am79C972 controller is not required to provide further controls or timing. The AMD Flash product will compliment EBD[7] during a read of the programmed location until the programming is complete. The host software should poll the programmed address until EBD[7] matches the programmed value.

AMD Flash byte programming is allowed in any sequence and across sector boundaries. *Note that a data 0 cannot be programmed back to a 1. Only erase operations can convert zeros to ones.* AMD Flash chip erase is a six-bus cycle operation. There are two *unlock* write cycles, followed by writing the set-up command. Two more *unlock* cycles are then followed by the chip erase command. Chip erase does *not* require the user to program the device prior to erasure. Upon executing the AMD Flash Embedded Erase Algorithm command sequence, the Flash device will program and verify the entire memory for an all zero data pattern prior to electrical erase. The Am79C972 controller is not required to provide any controls or timings during these operations. The automatic erase begins on the rising edge of the last $\overline{\text{EBWE}}$ pulse in the command sequence and terminates when the data on EBD[7] is 1, at which time the Flash device returns to the read mode. Polling by the Am79C972 controller is not required during the erase sequence. The following FLASH programming-table excerpt (Table 9) shows the command sequence for byte programming and sector/chip erasure on an AMD Flash device. In the following table, PA and PD stand for programmed address and programmed data, and SA stands for sector address.

The Am79C972 controller will support only a single sector erase per command and not concurrent sector erasures. The Am79C972 controller will support most FLASH devices as long as there is no timing requirement between the completion of commands. The FLASH access time cannot be guaranteed with the Am79C972 controller access mechanism. The Am79C972 controller will also support only Flash devices that do not require data hold times after write operations.

Table 9. Am29Fxxx Flash Command

Command Sequence	Bus Write Cycles Req'd	First Bus Write Cycle		Second Bus Write Cycle		Third Bus Write Cycle		Fourth Bus Write Cycle		Fifth Bus Write Cycle		Sixth Bus Write Cycle	
		Addr	Data	Addr	Data	Addr	Data	Addr	Data	Addr	Data	Addr	Data
Byte Program	4	5555h	AAh	2AAAh	55H	5555h	A0h	PA	PD				
Chip Erase	6	5555h	AAh	2AAAh	55H	5555h	80h	5555h	AAh	2AAAh	55h	5555h	10h
Sector Erase	6	5555h	AAh	2AAAh	55H	5555h	80h	5555h	AAh	2AAAh	55h	SA	3h

SRAM Configuration

The Am79C972 controller supports SRAM as a FIFO extension as well as providing a read/write data path to the SRAM. The Am79C972 controller contains 12 Kbytes of SRAM.

Internal SRAM Configuration

The SRAM_SIZE (BCR25, bits 7-0) programs the size of the SRAM. SRAM_SIZE can be programmed to a smaller value than 12 Kbytes.

The SRAM should be programmed on a 512-byte boundary. However, there should be no accesses to the RAM space while the Am79C972 controller is running. The Am79C972 controller assumes that it completely owns the SRAM while it is in operation. To specify how much of the SRAM is allocated to transmit and how much is allocated to receive, the user should program SRAM_BND (BCR26, bits 7-0) with the page boundary where the receive buffer begins. The SRAM_BND also should be programmed on a 512-byte boundary. The transmit buffer space starts at 0000h. It is up to the user or the software driver to split up the memory for transmit or receive; there is no defaulted value. The minimum SRAM size required is four 512-byte pages for each transmit and receive queue, which limits the SRAM size to be at least 4 Kbytes.

The SRAM_BND upon H_RESET will be reset to 0000h. The Am79C972 controller will not have any transmit buffer space unless SRAM_BND is programmed. The last configuration parameter necessary is the clock source used to control the Expansion Bus interface. This is programmed through the SRAM Interface Control register. The externally driven Expansion Bus Clock (EBCLK) can be used by specifying a value of 010h in EBCS (BCR27, bits 5-3). This allows the user to utilize any clock that may be available.

There are two standard clocks that can be chosen as well, the PCI clock or the externally provided time base clock. Use of the internal clock is not recommended. When the PCI or time base clock is used, the EBCLK does not have to be driven, but it must be tied to VDD through a resistor. The user must specify an SRAM clock (BCR27, bits 5-3) that will not stop unless the Am79C972 controller is stopped. Otherwise, the Am79C972 controller will report buffer overflows, underflows, corrupt data, and will hang eventually.

The user can decide to use a fast clock and then divide down the frequency to get a better duty-cycle if required. The choices are a divide by 2 or 4 and is programmed by the CLK_FAC bits (BCR27, bits 2-0). Note that the Am79C972 controller does not support an SRAM frequency above 33 MHz regardless of the clock and clock factor used.

No SRAM Configuration

If the SRAM_SIZE (BCR25, bits 7-0) value is 0 in the SRAM size register, the Am79C972 controller will assume that there is no SRAM present and will reconfigure the four internal FIFOs into two FIFOs, one for transmit and one for receive. The FIFOs will operate the same as in the PCnet-PCI II controller. When the SRAM SIZE (BCR25, bits 7-0) value is 0, the SRAM BND (BCR26, bits 7-0) are ignored by the Am79C972 controller. See Figure 45.

Low Latency Receive Configuration

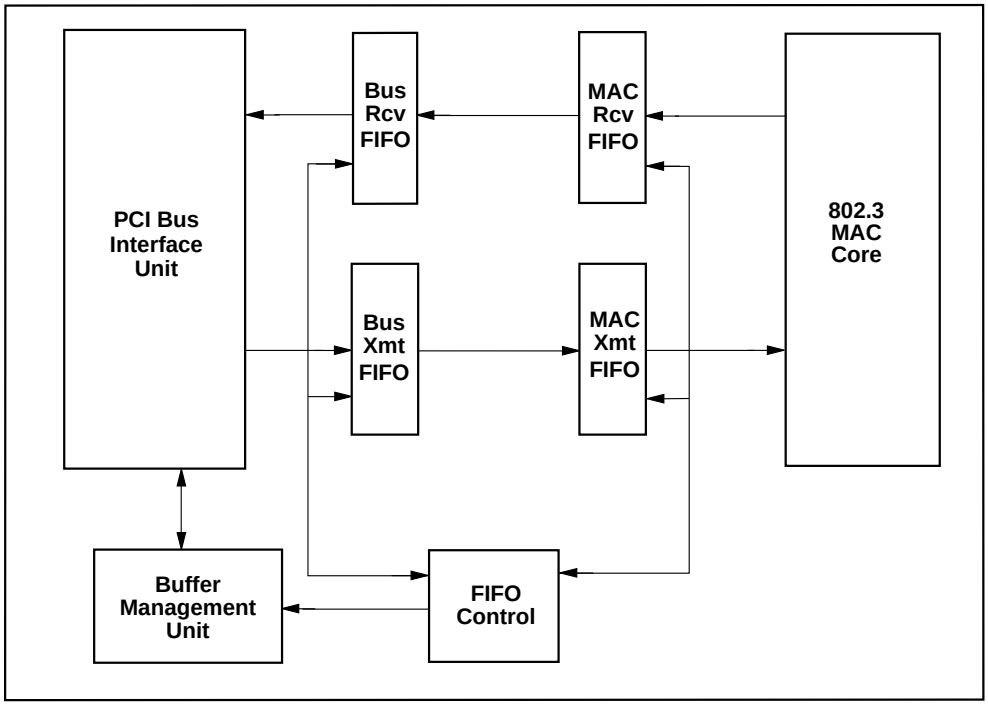
If the LOLATRX (BCR27, bit 4) bit is set to 1, then the Am79C972 controller will configure itself for a low latency receive configuration. In this mode, SRAM is required at all times. If the SRAM_SIZE (BCR25, bits 7-0) value is 0, the Am79C972 controller will not configure for low latency receive mode. The Am79C972 controller will provide a fast path on the receive side bypassing the SRAM. All transmit traffic will go to the SRAM, so SRAM_BND (BCR26, bits 7-0) has no meaning in low latency receive mode. When the Am79C972 controller has received 16 bytes from the network, it will start a DMA request to the PCI Bus Interface Unit. The Am79C972 controller will not wait for the first 64 bytes to pass to check for collisions in Low Latency Receive mode. The Am79C972 controller must be in STOP before switching to this mode. See Figure 46.

CAUTION: To provide data integrity when switching into and out of the low latency mode, DO NOT SET the FASTSPNDE bit when setting the SPND bit. Receive frames WILL be overwritten and the Am79C972 controller may give erratic behavior when it is enabled again.

Direct SRAM Access

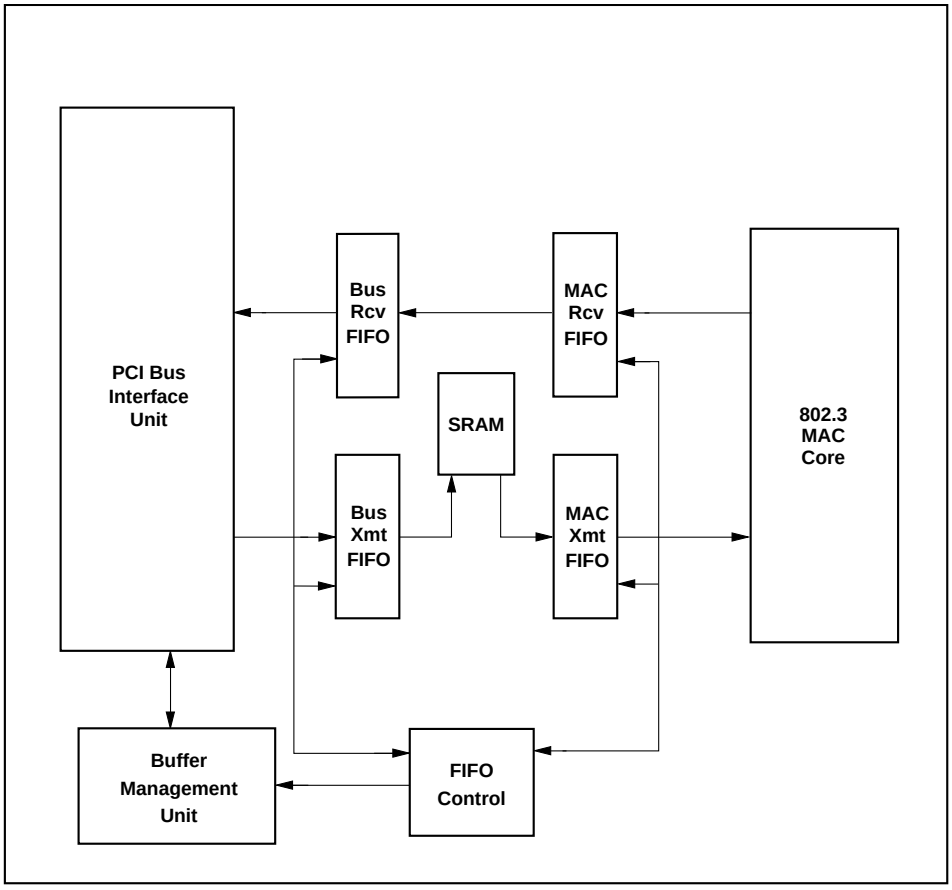
The SRAM can be accessed through the Expansion Bus Data port (BCR30). To access this data port, the user must load the upper address EPADDRU (BCR29, bits 3-0) and set FLASH (BCR29, bit 15) to 0. Then the user will load the lower 16 bits of address EPADDRL (BCR28, bits 15-0). To initiate a read, the user reads the Expansion Bus Data Port (BCR30). This slave access from the PCI will result in a retry for the very first access. Subsequent accesses may give a retry or not, depending on whether or not the data is present and valid. The direct SRAM access uses the same FLASH/EPROM access except for accessing the SRAM in word format instead of byte format. This access is meant to be a diagnostic access only. The SRAM can only be accessed while the Am79C972 controller is in STOP or SPND (FASTSPNDE is set to 0) mode.

.



21485C-48

Figure 45. Block Diagram No SRAM Configuration



21485C-49

Figure 46. Block Diagram Low Latency Receive Configuration

EEPROM Interface

The Am79C972 controller contains a built-in capability for reading and writing to an external serial 93C46 EEPROM. This built-in capability consists of an interface for direct connection to a 93C46 compatible EEPROM, an automatic EEPROM read feature, and a user-programmable register that allows direct access to the interface pins.

Automatic EEPROM Read Operation

Shortly after the deassertion of the $\overline{RST}$ pin, the Am79C972 controller will read the contents of the EEPROM that is attached to the interface. Because of this automatic-read capability of the Am79C972 controller, an EEPROM can be used to program many of the features of the Am79C972 controller at power-up, allowing system-dependent configuration information to be stored in the hardware, instead of inside the device driver.

If an EEPROM exists on the interface, the Am79C972 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 Am79C972 controller. Access to the Am79C972 configuration space, the Expansion ROM or any I/O resource is not possible during the EEPROM read operation. The Am79C972 controller will terminate any access attempt with the assertion of $\overline{DEVSEL}$ and $\overline{STOP}$ while $\overline{TRDY}$ is not asserted, signaling to the initiator to disconnect and 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 passes, PVALID (BCR19, bit 15) will be set to 1. If the checksum verification of the EEPROM data fails, PVALID will be cleared to 0, and the Am79C972 controller will force all EEPROM-programmable BCR registers back to their H_RESET default values. However, the content of the Address PROM locations (offsets 0h - Fh from the I/O or memory mapped I/O base address) will not be cleared. The 8-bit checksum for the entire 68 bytes of the EEPROM should be FFh.

If no EEPROM is present at the time of the automatic read operation, the Am79C972 controller will recognize this condition and will abort the automatic read operation and clear 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 or memory mapped I/O base address) will be undefined.

EEPROM Auto-Detection

The Am79C972 controller uses the EESK/$\overline{LED1}$/SFBD pin to determine if an EEPROM is present in the system. At the rising edge of CLK during the last clock during which $\overline{RST}$ is asserted, the Am79C972 controller will sample the value of the EESK/$\overline{LED1}$/SFBD pin. If the sampled value is a 1, then the Am79C972 controller assumes that an EEPROM is present, and the EEPROM read operation begins shortly after the $\overline{RST}$ pin is deasserted. If the sampled value of EESK/$\overline{LED1}$/SFBD is a 0, the Am79C972 controller assumes that an external pulldown device is holding the EESK/$\overline{LED1}$/SFBD pin low, indicating that there is no EEPROM in the system. Note that if the designer creates a system that contains an LED circuit on the EESK/$\overline{LED1}$/SFBD 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 Am79C972 controller, since the checksum verification will fail.

Direct Access to the Interface

The user may directly access the port through the EEPROM register, BCR19. This register contains bits that can be used to control the interface pins. By performing an appropriate sequence of 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:

- I/O offsets 0h-Fh Address PROM locations
- BCR2 Miscellaneous Configuration
- BCR4 LED0 Status
- BCR5 LED1 Status
- BCR6 LED2 Status
- BCR7 LED3 Status
- BCR9 Full-Duplex Control
- BCR18 Burst and Bus Control
- BCR22 PCI Latency
- BCR23 PCI Subsystem Vendor ID
- BCR24 PCI Subsystem ID
- BCR25 SRAM Size
- BCR26 SRAM Boundary
- BCR27 SRAM Interface Control
- BCR32 MII Control and Status
- BCR33 MII Address
- BCR35 PCI Vendor ID
- BCR36 PCI Power Management Capabilities (PMC) Alias Register

- BCR37 PCI DATA Register Zero (DATA0) Alias Register
- BCR38 PCI DATA Register One (DATA1) Alias Register
- BCR39 PCI DATA Register Two (DATA2) Alias Register
- BCR40 PCI DATA Register Three (DATA3) Alias Register
- BCR41 PCI DATA Register Four (DATA4) Alias Register
- BCR42 PCI DATA Register Five (DATA5) Alias Register
- BCR43 PCI DATA Register Six (DATA6) Alias Register
- BCR44 PCI DATA Register Seven (DATA7) Alias Register
- BCR45 OnNow Pattern Matching Register 1
- BCR46 OnNow Pattern Matching Register 2
- BCR47 OnNow Pattern Matching Register 3
- CSR116 OnNow Miscellaneous

If PREAD (BCR19, bit 14) and PVALID (BCR19, bit 15) are cleared to 0, 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 Address PROM locations, however, will not be cleared.

Accesses to the Address PROM 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 Am79C972 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 34 words (i.e., 68 bytes) of the EEPROM. The format of the EEPROM contents is shown in Table 10 (next page), beginning with the byte that resides at the lowest EEPROM address.

Note: *The first bit out of any word location in the EEPROM is treated as the MSB of the register being programmed. For example, the first bit out of EEPROM word location 09h will be written into BCR4, bit 15; the second bit out of EEPROM word location 09h will be written into BCR4, bit 14, etc.*

There are two checksum locations within the EEPROM. The first 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 43h) 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 68 bytes of EEPROM data equals the value FFh. The checksum adjust byte is needed by the Am79C972 controller in order to verify that the EEPROM content has not been corrupted.

LED Support

The Am79C972 controller can support up to four LEDs. LED outputs $\overline{\text{LED0}}$, $\overline{\text{LED1}}$, and $\overline{\text{LED2}}$ allow for direct connection of an LED and its supporting pullup device.

In applications that want to use the pin to drive an LED and also have an EEPROM, it might be necessary to buffer the $\overline{\text{LED3}}$ circuit from the EEPROM connection. When an LED circuit is directly connected to the EEDO/$\overline{\text{LED3}}$/SRD pin, then it is not possible for most EEPROM devices to sink enough I_{OL} to maintain a valid low level on the EEDO input to the Am79C972 controller. Use of buffering can be avoided if a low power LED is used.

Each LED can be programmed through a BCR register to indicate one or more of the following network status or activities: Collision Status, Full-Duplex Link Status, Half-Duplex Link Status, Receive Match, Receive Status, Magic Packet, Disable Transceiver, and Transmit Status.

Table 10. EEPROM Map

Word Address	Byte Address	Most Significant Byte	Byte Address	Least Significant Byte
00h*	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	CSR116[15:8] (OnNow Misc. Config.)	06h	CSR116[7:0] (OnNow Misc. Config.)
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	MSB of two-byte checksum, which is the sum of bytes 00h-0Bh and bytes 0Eh and 0Fh	0Ch	LSB 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	BCR2[15:8] (Miscellaneous Configuration)	10h	BCR2[7:0] (Miscellaneous Configuration)
09h	13h	BCR4[15:8] (Link Status LED)	12h	BCR4[7:0] (Link Status LED)
0Ah	15h	BCR5[15:8] (LED1 Status)	14h	BCR5[7:0] (LED1 Status)
0Bh	17h	BCR6[15:8] (LED2 Status)	16h	BCR6[7:0] (LED2 Status)
0Ch	19h	BCR7[15:8] (LED3 Status)	18h	BCR7[7:0] (LED3 Status)
0Dh	1Bh	BCR9[15:8] (Full-Duplex Control)	1Ah	BCR9[7:0] (Full-Duplex Control)
0Eh	1Dh	BCR18[15:8] (Burst and Bus Control)	1Ch	BCR18[7:0] (Burst and Bus Control)
0Fh	1Fh	BCR22[15:8] (PCI Latency)	1Eh	BCR22[7:0] (PCI Latency)
10h	21h	BCR23[15:8] (PCI Subsystem Vendor ID)	20h	BCR23[7:0] (PCI Subsystem Vendor ID)
11h	23h	BCR24[15:8] (PCI Subsystem ID)	22h	BCR24[7:0] (PCI Subsystem ID)
12h	25h	BCR25[15:8] (SRAM Size)	24h	BCR25[7:0] (SRAM Size)
13h	27h	BCR26[15:8] (SRAM Boundary)	26h	BCR26[7:0] (SRAM Boundary)
14h	29h	BCR27[15:8] (SRAM Interface Control)	28h	BCR27[7:0] (SRAM Interface Control)
15h	2Bh	BCR32[15:8] (MII Control and Status)	2Ah	BCR32[7:0] (MII Control and Status)
16h	2Dh	BCR33[15:8] (MII Address)	2Ch	BCR33[7:0] (MII Address)
17h	2Fh	BCR35[15:8] (PCI Vendor ID)	2Eh	BCR35[7:0] (PCI Vendor ID)
18h	31h	BCR36[15:8] (Conf. Space byte 43h alias)	30h	BCR36[7:0] (Conf. Space byte 42h alias)
19h	33h	BCR37[15:8] (DATA_SCALE alias 0)	32h	BCR37[7:0] (Conf. Space byte 47h 0 alias)
1Ah	35h	BCR38[15:8] (DATA_SCALE alias 1)	34h	BCR38[7:0] (Conf. Space. byte 47h 1 alias)
1Bh	75h	BCR39[15:8] (DATA_SCALE alias 2)	36h	BCR39[7:0] (Conf. Space. byte 47h 2 alias)
1Ch	39h	BCR40[15:8] (DATA_SCALE alias 3)	38h	BCR40[7:0] (Conf. Space. byte 47h 3 alias)
1Dh	3Bh	BCR41[15:8] (DATA_SCALE alias 4)	3Ah	BCR41[7:0] (Conf. Space. byte 47h 4 alias)
1Eh	3Dh	BCR42[15:8] (DATA_SCALE alias 0)	3Ch	BCR42[7:0] (Conf. Space. byte 47h 5 alias)
1Fh	3Fh	BCR43[15:8] (DATA_SCALE alias 0)	3Eh	BCR43[7:0] (Conf. Space. byte 47h 6 alias)
20h	41h	BCR44[15:8] (DATA_SCALE alias 0)	40h	BCR44[7:0] (Conf. Space. byte 47h 7 alias)
21h	43h	Checksum adjust byte for the 68 bytes of the EEPROM contents. Checksum of the 68 bytes of the EEPROM should total FFh.	42h	Reserved location: must be 00h
Unused locations - Ignored by device				
3Fh	7Fh	Reserved	7Eh	Reserved

The LED pins can be configured to operate in either open-drain mode (active low) or in totem-pole mode (active high). The output can be stretched to allow the human eye to recognize even short events that last only several microseconds. After H_RESET, the four LED outputs are configured as shown in Table 11.

Table 11. LED Default Configuration

LED Output	Indication	Driver Mode	Pulse Stretch
LED0	Link Status	Open Drain - Active Low	Enabled
LED1	Receive Status	Open Drain - Active Low	Enabled
LED2	--	Open Drain - Active Low	Enabled
LED3	Transmit Status	Open Drain - Active Low	Enabled

For each LED register, each of the status signals is AND'd with its enable signal, and these signals are all OR'd together to form a combined status signal. Each LED pin combined status signal can be programmed to run 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 to 3 clocks of stretched LED output, or 52 ms to 78 ms. See Figure 47.

Power Savings Mode

Power Management Support

PCnet-FAST+ supports power management as defined in the PCI Bus Power Management Interface Specification V1.0 and Network Device Class Power Management Reference Specification V1.0.These specifications define the network device power states, PCI power management interface including the Capabilities Data Structure and power management registers block definitions, power management events, and OnNow network Wake-up events. In addition, PCnet-*FAST*+ supports legacy power management schemes, such as Remote Wake-Up (RWU) mode. When the system is in RWU mode, PCI bus power is on, the PCI clock may be slowed down or stopped, and the wake-up output pin may drive the CPU's System Management Interrupt (SMI) line.

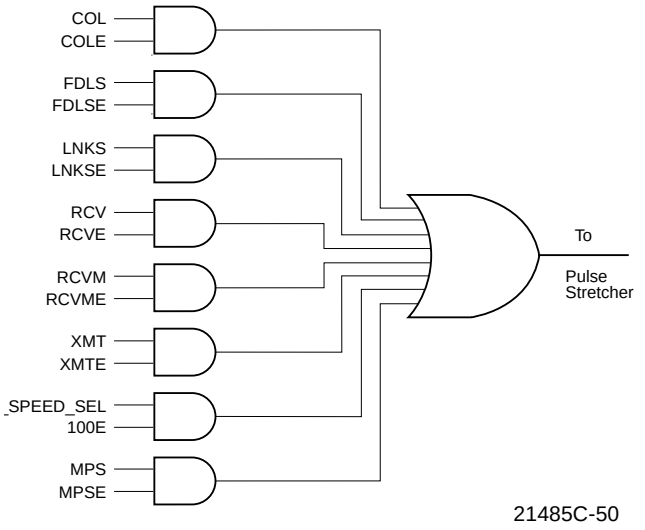


Figure 47. LED Control Logic

The general scheme for the PCnet-*FAST*+ power management is that when a PCI Wake-up event is detected, a signal is generated to cause hardware external to the PCnet-*FAST*+ device to put the computer into the working (S0) mode.

The PCnet-*FAST*+ device supports three types of wake-up events:

1. Magic Packet Detect
2. OnNow Pattern Match Detect
3. Link State Change

Figure 48 shows the relationship between these Wake-up events and the various outputs used to signal to the external hardware.

Note: *The OnNOW Pattern Match and Link State Change only work on the MII interface.*

OnNow Wake-Up Sequence

The system software enables the $\overline{\text{PME}}$ pin by setting the PME_EN bit in the PMCSR register (PCI configuration registers, offset 44h, bit 8) to 1. When a Wake-up event is detected, the PCnet-*FAST*+ device sets the PME_STATUS bit in the PMCSR register (PCI configuration registers, offset 44h, bit 15). Setting this bit causes the $\overline{\text{PME}}$ signal to be asserted. Assertion of the $\overline{\text{PME}}$ signal causes external hardware to wake up the CPU. The system software then reads the PMCSR register of every PCI device in the system to determine which device asserted the $\overline{\text{PME}}$ signal.

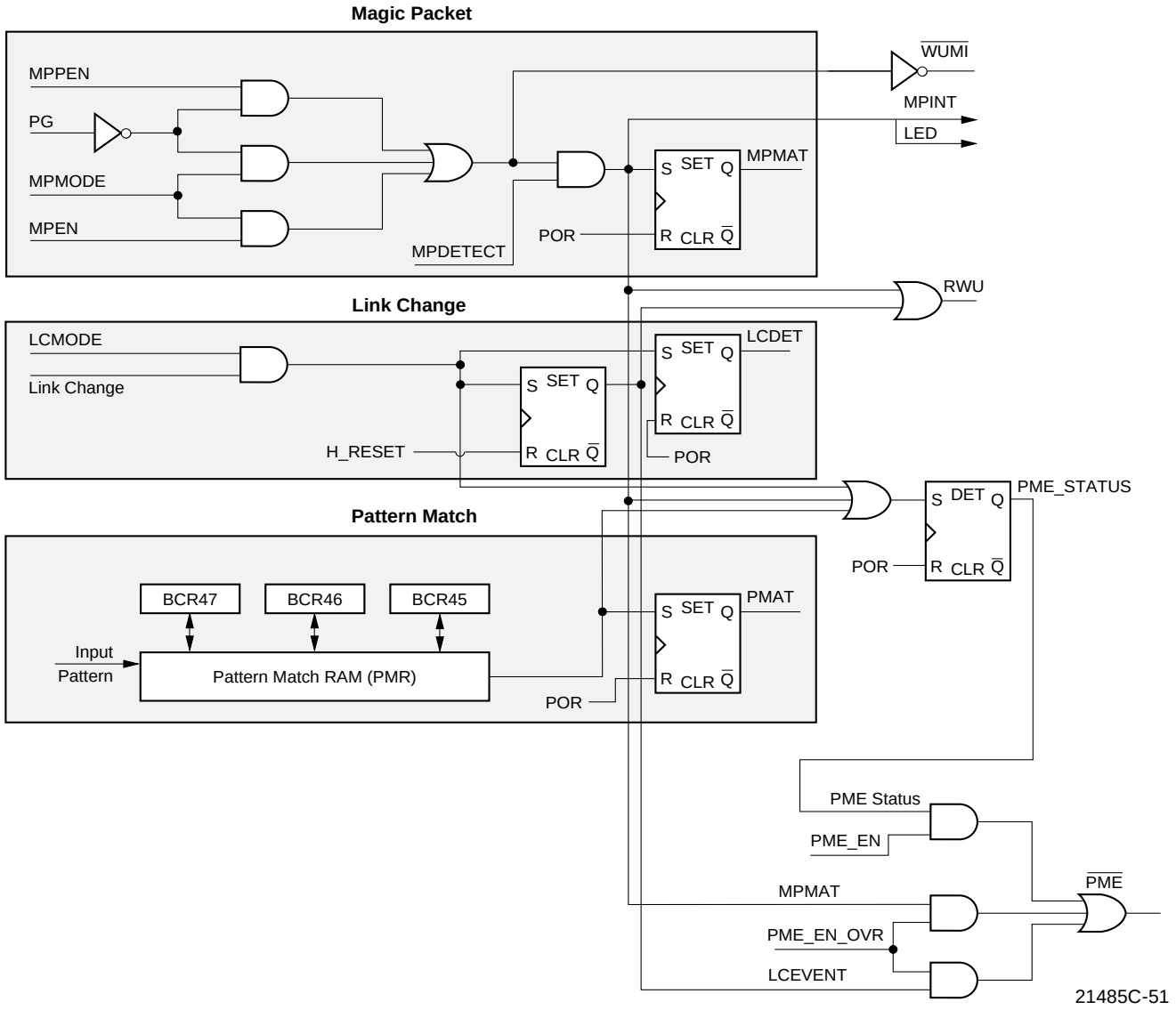


Figure 48. OnNow Functional Diagram

When the software determines that the signal came from the PCnet-*FAST*+ device, it writes to the device's PMCSR to put the device into power state D0. The software then writes a 0 to the PME_STATUS bit to clear the bit and turn off the $\overline{\text{PME}}$ signal, and it calls the device's software driver to tell it that the device is now in state D0. The system software can clear the PME_STATUS bit either before, after, or at the same time that it puts the device back into the D0 state.

Link Change Detect

Link change detect is one of Wake-up events defined by the OnNow specification and is supported by the RWU mode. Link Change Detect mode is set when the LCMODE bit (CSR116, bit 8) is set either by software or loaded through the EEPROM.

When this bit is set, any change in the Link status will cause the LCDET bit (CSR116, bit 9) to be set. When the LCDET bit is set, the RWU pin will be asserted and the PME_STATUS bit (PMCSR register, bit 15) will be set. If either the PME_EN bit (PMCSR, bit 8) or the PME_EN_OVR bit (CSR116, bit 10) are set, then the $\overline{\text{PME}}$ will also be asserted.

OnNow Pattern Match Mode

In the OnNow Pattern Match Mode, the PCnet-*FAST*+ compares the incoming packets with up to eight patterns stored in the Pattern Match RAM (PMR). The stored patterns can be compared with part or all of incoming packets, depending on the pattern length and the way the PMR is programmed. When a pattern match has been detected, then PMAT bit (CSR116, bit 7) is set. The setting of the PMAT bit causes the

PME_STATUS bit (PMCSR, bit 15) to be set, which in turn will assert the $\overline{\text{PME}}$ pin if the PME_EN bit (PMCSR, bit 8) is set.

Pattern Match RAM (PMR)

PMR is organized as an array of 64 words by 40 bits as shown in Figure 49. The PMR is programmed indirectly through the BCRs 45, 46, and 47. When the BCR45 is written and the PMAT_MODE bit (BCR45, bit 7) is set to 1, Pattern Match logic is enabled. No bus accesses into the PMR are possible when the PMAT_MODE bit is set, and BCR46, BCR47, and all other bits in BCR45 are ignored. When PMAT_MODE is set, a read of BCR45 returns all bits undefined except for PMAT_MODE. In order to access the contents of the PMR, PMAT_MODE bit should be programmed to 0.

When BCR45 is written to set the PMAT_MODE bit to 0, the Pattern Match logic is disabled and accesses to the PMR are possible. Bits 6:0 of BCR45 specify the address of the PMR word to be accessed. Writing to BCR45 does not immediately affect the contents of the PMR. Following the write to BCR45, the PMR word addressed by the bits 6:0 of the BCR45 may be read by reading BCR45, BCR46, and BCR47 in any order. To write to the PMR word, the write to BCR45 must be followed by a write to BCR46 and a write to BCR47 in that order to complete the operation. The PMR will not actually be written until the write to BCR47 is complete.

The first two 40-bit words in this RAM serve as pointers and contain enable bits for the eight possible match patterns. The remainder of the RAM contains the match patterns and associated match pattern control bits. The byte 0 of the first word contains the Pattern Enable bits. Any bit position set in this byte enables the corresponding match pattern in the PMR, as an example if the bit 3 is set, then the Pattern 3 is enabled for matching. Bytes 1 to 4 in the first word are pointers to the beginning of the patterns 0 to 3, and bytes 1 to 4 in the second word are pointers to the beginning of the patterns 4 to 7, respectively. Byte 0 of the second word has no function associated with it.The byte 0 of the words 2 to 63 is the Control Field of the PMR. Bit 7 of this field is the End of Packet (EOP) bit. When this bit is set, it indicates the end of a pattern in the PMR. Bits 6-4 of the Control Field byte are the SKIP bits. The value of the SKIP field indicates the number of the Dwords to be skipped before the pattern in this PMR word is compared with data from the incoming frame. A maximum of seven Dwords may be skipped. Bits 3-0 of the Control Field byte are the MASK bits. These bits correspond to the pattern match bytes 3-0 of the same PMR word (PMR bytes 4-1). If bit *n* of this field is 0, then byte *n* of the corresponding pattern word is ignored. If this field is programmed to 3, then bytes 0 and 1 of the pattern match field (bytes 2 and 1 of the word) are used and bytes 3 and 2 are ignored in the pattern matching operation.

The contents of the PMR are not affected by H_RESET, S_RESET, or STOP. The contents are undefined after a power up reset (POR).

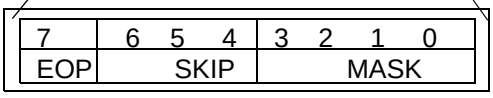
Magic Packet Mode

In Magic Packet mode, the PCnet-*FAST*+ controller remains fully powered up (all VDD and VDDB pins must remain at their supply levels). The device will not generate any bus master transfers. No transmit operations will be initiated on the network. The device will continue to receive frames from the network, but all frames will be automatically flushed from the receive FIFO. Slave accesses to the PCnet-*FAST*+ controller are still possible. A Magic Packet is a frame that is addressed to the PCnet-*FAST*+ controller and contains a data sequence anywhere in its data field made up of 16 consecutive copies of the device's physical address (PADR[47:0]). The PCnet-*FAST*+ controller will search incoming frames until it finds a Magic Packet frame. It starts scanning for the sequence after processing the length field of the frame. The data sequence can begin anywhere in the data field of the frame, but must be detected before the PCnet-*FAST*+ controller reaches the frame's FCS field. Any deviation of the incoming frame's data sequence from the required physical address sequence, even by a single bit, will prevent the detection of that frame as a Magic Packet frame.

The PCnet-*FAST*+ controller supports two different modes of address detection for a Magic Packet frame. If MPPLBA (CSR5, bit 5) or EMPPLBA (CSR116, bit 6) are at their default value of 0, the PCnet-*FAST*+ controller will only detect a Magic Packet frame if the destination address of the packet matches the content of the physical address register (PADR). If MPPLBA or EMPPLBA are set to 1, the destination address of the Magic Packet frame can be unicast, multicast, or broadcast.

Note: *The setting of MPPLBA or EMPPLBA only effects the address detection of the Magic Packet frame. The Magic Packet's data sequence must be made up of 16 consecutive copies of the device's physical address (PADR[47:0]), regardless of what kind of destination address it has.*

	BCR 47		BCR 46		BCR 45	
BCR Bit Number	15 8	7 0	15 8	7 0	15 8	
	PMR_B4	**PMR_B3**	**PMR_B2**	**PMR_B1**	**PMR_B0**	
Pattern Match RAM Address	Pattern Match RAM Bit Number 39 32	31 24	23 16	15 8	7 0	Comments
0	P3 pointer	P2 pointer	P1 pointer	P0 pointer	Pattern Enable bits	First Address
1	P7 pointer	P6 pointer	P5 pointer	P4 pointer	X	Second Address
2	Data Byte 3	Data Byte 2	Data Byte1	Data Byte 0	Pattern Control	Start Pattern P_1
2+n	Data Byte 4n+3	Date Byte 4n+2	Data Byte 4n+1	Data Byte 4n+0	Pattern Control	End Pattern P_1
J	Data Byte 3	Data Byte 2	Data Byte 1	Data Byte 0	Pattern Control	Start Pattern P_k
J+m	Data Byte 4m+3	Data Byte 4m+2	Data Byte 4m+1	Data Byte 4m+0	Pattern Control	End Pattern P_k
63						Last Address



21485C-52

Figure 49. Pattern Match RAM

There are two general methods to place the PCnet-*FAST*+ into the Magic Packet mode. The first is the software method. In this method, either the BIOS or other software, sets the MPMODE bit (CSR5, bit 1). Then PCnet-*FAST*+ controller must be put into suspend mode (see description of CSR5, bit 0), allowing any current network activity to finish. Finally, either PG must be deasserted (hardware control) or MPEN (CSR5, bit 2) must be set to 1 (software control).

Note: *FASTSPNDE (CSR7, bit 15) has no meaning in Magic Packet mode.*

The second method is the hardware method. In this method, the MPPEN bit (CSR116, bit 4) is set at power up by the loading of the EEPROM. This bit can also be set by software. The PCnet-*FAST*+ will be placed in the Magic Packet Mode when either the PG input is deasserted or the MPEN bit is set. $\overline{\text{WUMI}}$ output will be asserted when the PCnet-*FAST*+ is in the Magic Packet mode. Magic Packet mode can be disabled at any time by asserting PG or clearing MPEN bit.

When the PCnet-*FAST*+ controller detects a Magic Packet frame, it sets the MPMAT bit (CSR116, bit 5), the MPINT bit (CSR5, bit 4), and the PME_STATUS bit (PMCSR, bit 15). The setting of the MPMAT bit will also cause the RWU pin to be asserted and if the PME_EN or the PME_EN_OVR bits are set, then the $\overline{\text{PME}}$ will be asserted as well. If IENA ($\overline{\text{CSR0}}$, bit 6) and MPINTE (CSR5, bit 3) are set to 1, $\overline{\text{INTA}}$ will be asserted. Any one of the four LED pins can be programmed to indicate that a Magic Packet frame has been received.

MPSE (BCR4-7, bit 9) must be set to 1 to enable that function.

Note: *The polarity of the LED pin can be programmed to be active HIGH by setting LEDPOL (BCR4-7, bit 14) to 1.*

Once a Magic Packet frame is detected, the PCnet-*FAST*+ controller will discard the frame internally, but will not resume normal transmit and receive operations until PG is asserted or MPEN is cleared. Once both of these events has occurred, indicating that the system has detected the Magic Packet and is awake, the controller will continue polling receive and transmit descriptor rings where it left off. It is not necessary to re-initialize the device. If the part is reinitialized, then the descriptor locations will be reset and the PCnet-*FAST*+ controller will not start where it left off.

If magic packet mode is disabled by the assertion of PG, then in order to immediately re-enable Magic Packet mode, the PG pin must remain asserted for at least 200 ns before it is deasserted. If Magic Packet mode is disabled by clearing MPEN bit, then it may be immediately re-enabled by setting MPEN back to 1.

The PCI bus interface clock (CLK) is not required to be running while the device is operating in Magic Packet mode. Either of the $\overline{\text{INTA}}$, the LED pins, RWU or the $\overline{\text{PME}}$ signal may be used to indicate the receipt of a Magic Packet frame when the CLK is stopped. If the system wishes to stop the CLK, it will do so after enabling the Magic Packet mode.

CAUTION: *To prevent unwanted interrupts from other active parts of the PCnet-FAST+ controller, care must be taken to mask all likely interruptible events during Magic Packet mode. An example would be the interrupts from the Media Independent Interface, which could occur while the device is in Magic Packet mode.*

IEEE 1149.1 (1990) Test Access Port Interface

An IEEE 1149.1-compatible boundary scan Test Access Port is provided for board-level continuity test and diagnostics. All digital input, output, and input/output pins are tested. The following paragraphs summarize the IEEE 1149.1-compatible test functions implemented in the Am79C972 controller.

Boundary Scan Circuit

The boundary scan test circuit requires four pins (TCK, TMS, TDI, and TDO), defined as the Test Access Port (TAP). It includes a finite state machine (FSM), an instruction register, a data register array, and a power-on reset circuit. Internal pull-up resistors are provided for the TDI, TCK, and TMS pins.

TAP Finite State Machine

The TAP engine is a 16-state finite state machine (FSM), driven by the Test Clock (TCK), and the Test Mode Select (TMS) pins. An independent power-on reset circuit is provided to ensure that the FSM is in the TEST_LOGIC_RESET state at power-up. Therefore, the $\overline{\text{TRST}}$ is not provided. The FSM is also reset when TMS and TDI are high for five TCK periods.

Supported Instructions

In addition to the minimum IEEE 1149.1 requirements (BYPASS, EXTEST, and SAMPLE instructions), three additional instructions (IDCODE, TRIBYP, and SETBYP) are provided to further ease board-level testing. All unused instruction codes are reserved. See Table 12 for a summary of supported instructions.

Table 12. IEEE 1149.1 Supported Instruction Summary

Instruction Name	Instruction Code	Description	Mode	Selected Data Register
EXTEST	0000	External Test	Test	BSR
IDCODE	0001	ID Code Inspection	Normal	ID REG
SAMPLE	0010	Sample Boundary	Normal	BSR
TRIBYP	0011	Force Float	Normal	Bypass
SETBYP	0100	Control Boundary To 1/0	Test	Bypass
BYPASS	1111	Bypass Scan	Normal	Bypass

Instruction Register and Decoding Logic

After the TAP FSM is reset, the IDCODE instruction is always invoked. The decoding logic gives signals to control the data flow in the Data registers according to the current instruction.

Boundary Scan Register

Each Boundary Scan Register (BSR) cell has two stages. A flip-flop and a latch are used for the Serial Shift Stage and the Parallel Output Stage, respectively. There are four possible operation modes in the BSR cell shown in Table 13.

Table 13. BSR Mode Of Operation

1	Capture
2	Shift
3	Update
4	System Function

Other Data Registers

Other data registers are the following:

1. Bypass Register (1 bit)
2. Device ID register (32 bits) (Table 14).

Table 14. Device ID Register

Bits 31-28	Version
Bits 27-12	Part Number (0010 0110 0010 0100)
Bits 11-1	Manufacturer ID. The 11 bit manufacturer ID cod for AMD is 00000000001 in accordance with JEDEC publication 106-A.
Bit 0	Always a logic 1

Note: *The content of the Device ID register is the same as the content of CSR88.*

NAND Tree Testing

The Am79C972 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, TBC_EN, and $\overline{EAR}$ pins.

NAND tree testing is enabled by asserting $\overline{RST}$. PG input should be driven HIGH during NAND tree testing. All PCI bus signals will become inputs on the assertion of $\overline{RST}$. The result of the NAND tree test can be observed on the $\overline{INTA}$ pin. See Figure 50.

Pin 143 ($\overline{RST}$) is the first input to the NAND tree. Pin 144 (CLK) is the second input to the NAND tree, followed by pin 145 ($\overline{GNT}$). All other PCI bus signals follow, counterclockwise, with pin 129 ($\overline{EAR}$) being the last. Table 15 shows the complete list of pins connected to the NAND tree.

$\overline{RST}$ 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 $\overline{INTA}$ 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, $\overline{INTA}$ will toggle every time an additional input is driven low. $\overline{INTA}$ will change to low, when CLK is driven low and all other NAND tree inputs stay high. $\overline{INTA}$ will toggle back to high, when $\overline{GNT}$ is additionally driven low. The square wave will continue until all NAND tree inputs are driven low. $\overline{INTA}$ will be high, when all NAND tree inputs are driven low. See Figure 51.

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 Am79C972 controller is configured for NAND tree testing.

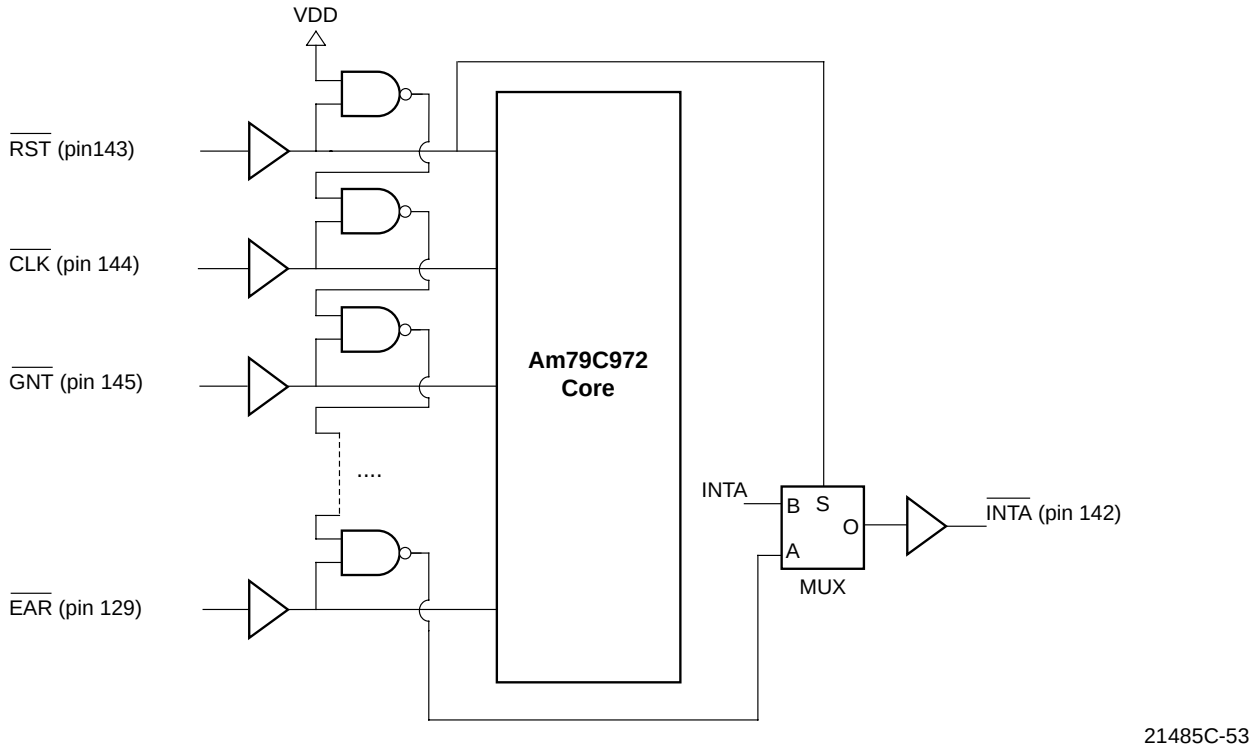


Figure 50. NAND Tree Circuitry

Table 15. NAND Tree Pin Sequence

NAND Tree Input No.	Pin No.	Name	NAND Tree Input No.	Pin No.	Name	NAND Tree Input No.	Pin No.	Name
1	143	$\overline{RST}$	18	7	AD20	35	34	AD13
2	144	CLK	19	9	AD19	36	36	AD12
3	145	$\overline{GNT}$	20	10	AD18	37	37	AD11
4	146	$\overline{REQ}$	21	12	AD17	38	39	AD10
5	148	AD31	22	14	AD16	39	40	AD9
6	151	AD30	23	15	C/$\overline{BE2}$	40	41	AD8
7	152	AD29	24	17	$\overline{FRAME}$	41	42	C/$\overline{BE0}$
8	153	AD28	25	18	$\overline{IRDY}$	42	44	AD7
9	154	AD27	26	20	$\overline{TRDY}$	43	46	AD6
10	156	AD26	27	22	$\overline{DEVSEL}$	44	47	AD5
11	158	AD25	28	23	$\overline{STOP}$	45	49	AD4
12	159	AD24	29	25	$\overline{PERR}$	46	50	AD3
13	160	C/$\overline{BE3}$	30	26	$\overline{SERR}$	47	52	AD2
14	1	IDSEL	31	28	PAR	48	54	AD1
15	2	AD23	32	30	C/$\overline{BE1}$	49	55	AD0
16	4	AD22	33	31	AD15	50	123	TBC_EN
17	6	AD21	34	33	AD14	51	129	$\overline{EAR}$

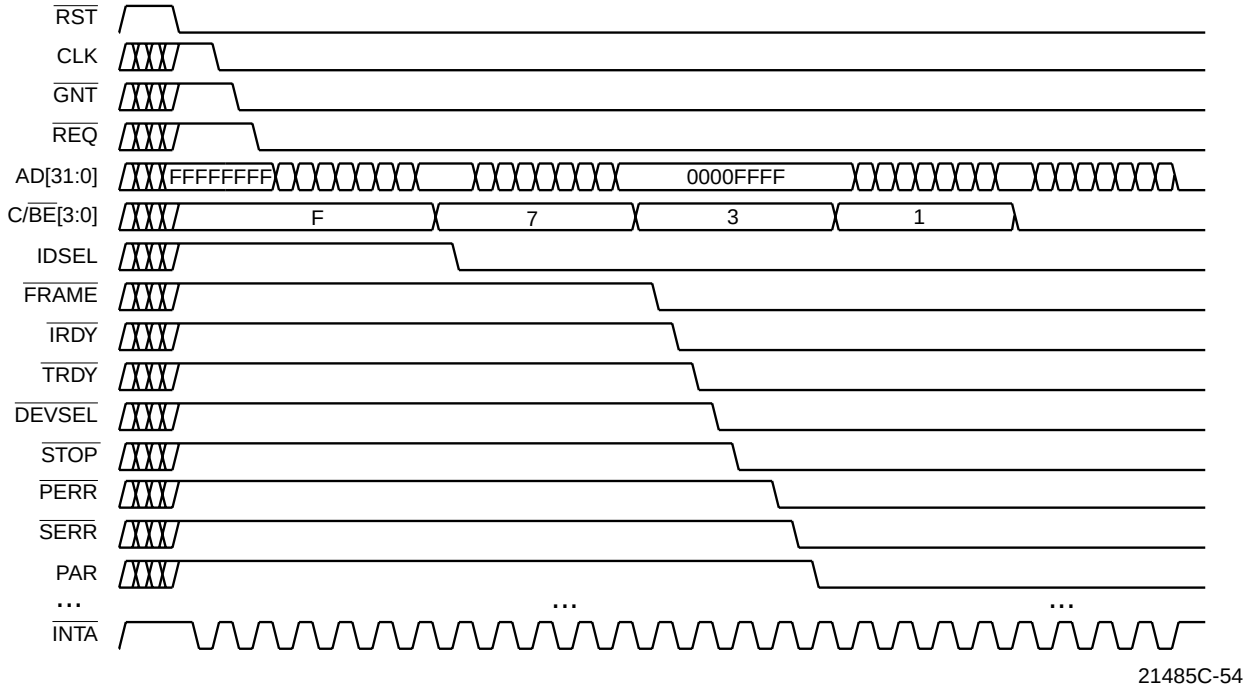


Figure 51. NAND Tree Waveform

Reset

There are four different types of RESET operations that may be performed on the Am79C972 device, H_RESET, S_RESET, STOP, and POR. The following is a description of each type of RESET operation.

H_RESET

Hardware Reset (H_RESET) is an Am79C972 reset operation that has been created by the proper assertion of the $\overline{RST}$ pin of the Am79C972 device while the PG pin is HIGH. When the minimum pulse width timing as specified in the $\overline{RST}$ pin description has been satisfied, then an internal reset operation will be performed.

H_RESET will program most of the CSR and BCR registers to their default value. Note that there are several CSR and BCR registers that are undefined after H_RESET. See the sections on the individual registers for details.

H_RESET will clear most of the registers 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 Am79C972 controller will attempt to read the EEPROM device through the EEPROM interface.

H_RESET will clear DWIO (BCR18, bit 7) and the Am79C972 controller will be in 16-bit I/O mode after the reset operation. A DWord write operation to the RDP (I/O offset 10h) must be performed to set the device into 32-bit I/O mode.

S_RESET

Software Reset (S_RESET) is an Am79C972 reset operation that has been created by a read access to the Reset register, which is located at offset 14h in Word I/O mode or offset 18h in DWord I/O mode from the Am79C972 I/O or memory mapped 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. For the identity of individual CSRs and bit locations that are affected by S_RESET, see the individual CSR register descriptions. S_RESET will not affect any PCI configuration space location. S_RESET will not affect any of the BCR register values. S_RESET will cause the microcode program to jump to its reset state. Following the end of the S_RESET operation, the Am79C972 controller will not attempt to read the EEPROM device. After S_RESET, the host must perform a full re-initialization of the Am79C972 controller before starting network activity. S_RESET will cause $\overline{REQ}$ to deassert immediately. STOP (CSR0, bit 2) or SPND (CSR5, bit 0) can be used to terminate any pending bus mastership request in an orderly sequence.

S_RESET terminates all network activity abruptly. The host can use the suspend mode (SPND, CSR5, bit 0) to terminate all network activity in an orderly sequence before issuing an S_RESET.

STOP

A STOP reset is generated by the assertion of the STOP bit in CSR0. Writing a 1 to the STOP bit of CSR0, when the stop bit currently has a value of 0, will initiate a STOP reset. If the STOP bit is already a 1, then writing a 1 to the STOP bit will not generate a STOP reset.

STOP will reset all or some portions of CSR0, 3, and 4 to default values. For the identity of individual CSRs and bit locations that are affected by STOP, see the individual CSR register descriptions. 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 Am79C972 controller will not attempt to read the EEPROM device.

Note: *STOP will not cause a deassertion of the $\overline{REQ}$ signal, if it happens to be active at the time of the write to CSR0. The Am79C972 controller will wait until it gains bus ownership and it will first finish all scheduled bus master accesses before the STOP reset is executed.*

STOP terminates all network activity abruptly. The host can use the suspend mode (SPND, CSR5, bit 0) to terminate all network activity in an orderly sequence before setting the STOP bit.

Power on Reset

Power on Reset (POR) is generated when the Am79C972 controller is powered up. POR generates a hardware reset (H_RESET). In addition, it clears some bits that H_RESET does not affect.

Software Access

PCI Configuration Registers

The Am79C972 controller implements the 256-byte configuration space as defined by the PCI specification revision 2.1. The 64-byte header includes all registers required to identify the Am79C972 controller and its function. Additionally, PCI Power Management Interface registers are implemented at location 40h - 47h. The layout of the Am79C972 PCI configuration space is shown in Table 16.

The PCI configuration registers are accessible only by configuration cycles. All multi-byte numeric fields follow little endian byte ordering. All write accesses to Reserved locations have no effect; reads from these locations will return a data value of 0.

I/O Resources

The Am79C972 controller requires 32 bytes of address space for access to all the various internal registers as well as to some setup information stored in an external serial EEPROM. A software reset port is available, too.

Table 16. PCI Configuration Space Layout

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
I/O Base Address				10h
Memory Mapped I/O Base Address				14h
Reserved				18h
Reserved				1Ch
Reserved				20h
Reserved				24h
Reserved				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved			CAP-PTR	34h
Reserved				38h
MAX_LAT	MIN_GNT	Interrupt Pin	Interrupt Line	3Ch
PMC		NXT_ITM_PTR	CAP_ID	40h
DATA_REG	PMCSR_BSE	PMCSR		44H
Reserved				. .
Reserved				FCh

The Am79C972 controller supports mapping the address space to both I/O and memory space. The value in the PCI I/O Base Address register determines the start address of the I/O address space. The 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 PCI Command register to enable I/O accesses to the Am79C972 controller. For memory mapped I/O access, the PCI Memory Mapped I/O Base Address register controls the start address of the memory space. The MEMEN bit in the PCI Command register must also be set to enable the mode. Both base address registers can be active at the same time.

The Am79C972 controller supports two modes for accessing the I/O resources. For backwards compatibility with AMD's 16-bit Ethernet controllers, Word I/O is the default mode after power up. The device can be configured to DWord I/O mode by software.

I/O Registers

The Am79C972 controller registers are divided into two groups. The Control and Status Registers (CSR) are used to configure the Ethernet MAC engine and to obtain status information. The Bus Control Registers (BCR) are used to configure the bus interface unit and the LEDs. Both sets of registers are accessed using indirect addressing.

The CSR and BCR share a common Register Address Port (RAP). There are, however, separate data ports. The Register Data Port (RDP) is used to access a CSR. The BCR Data Port (BDP) is used to access a BCR.

In order to access a particular CSR location, the RAP should first be written with the appropriate CSR address. The RDP will then point to the selected CSR. A read of the RDP will yield the selected CSR data. A write to the RDP will write to the selected CSR. In order to access a particular BCR location, the RAP should first be written with the appropriate BCR address. The BDP will then point to the selected BCR. A read of the BDP will yield the selected BCR data. A write to the BDP will write to the selected BCR.

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 cleared to all 0s when an H_RESET or S_RESET occurs. RAP is unaffected by setting the STOP bit.

Address PROM Space

The Am79C972 controller allows for connection of a serial EEPROM. The first 16 bytes of the EEPROM will be automatically loaded into the Address PROM (APROM) space after H_RESET. Additionally, the first six bytes of the EEPROM will be loaded into CSR12 to CSR14. The Address PROM space is a convenient place to store the value of the 48-bit IEEE station address. It can be overwritten by the host computer and its content has no effect on the operation of the controller. The software must copy the station address from the Address PROM space to the initialization block in

order for the receiver to accept unicast frames directed to this station.

The six bytes of the IEEE station address occupy the first six locations of the Address PROM 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. APROMWE bit (BCR2, bit 8) must be set to 1 to enable write access to the Address PROM space.

Reset Register

A read of the Reset register creates an internal software reset (S_RESET) pulse in the Am79C972 controller. 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, S_RESET is the equivalent of the assertion of the $\overline{RST}$ pin (H_RESET) except that S_RESET has no effect on the BCR or PCI Configuration space locations.

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 Am79C972 controller does not have a similar requirement. The write access is not required and does not have any effect.

Note: *The Am79C972 controller cannot service any slave accesses for a very short time after a read access of the Reset register, because the internal S_RESET operation takes about 1 µs to finish. The Am79C972 controller will terminate all slave accesses with the assertion of $\overline{DEVSEL}$ and $\overline{STOP}$ while $\overline{TRDY}$ is not asserted, signaling to the initiator to disconnect and retry the access at a later time.*

Word I/O Mode

After H_RESET, the Am79C972 controller is programmed to operate in Word I/O mode. DWIO (BCR18, bit 7) will be cleared to 0. Table 17 shows how the 32 bytes of address space are used in Word I/O mode.

All I/O resources must be accessed in word quantities and on word addresses. The Address PROM locations can also be read in byte quantities. The only allowed DWord operation is a write access to the RDP, which switches the device to DWord I/O mode. A read access other than listed in the table below will yield undefined data, a write operation may cause unexpected reprogramming of the Am79C972 control registers. Table 18 shows legal I/O accesses in Word I/O mode.

Table 17. I/O Map In Word I/O Mode (DWIO = 0)

Offset	No. of Bytes	Register
00h - 0Fh	16	APROM
10h	2	RDP
12h	2	RAP (shared by RDP and BDP)
14h	2	Reset Register
16h	2	BDP
18h - 1Fh	8	Reserved

Double Word I/O Mode

The Am79C972 controller can be configured to operate in DWord (32-bit) I/O mode. The software can invoke the DWIO mode by performing a DWord write access to the I/O location at offset 10h (RDP). The data of the write access must be such that it does not affect the intended operation of the Am79C972 controller. Setting the device into 32-bit I/O mode is usually the first operation after H_RESET or S_RESET. The RAP register will point to CSR0 at that time. Writing a value of 0 to CSR0 is a safe operation. DWIO (BCR18, bit 7) will be set to 1 as an indication that the Am79C972 controller operates in 32-bit I/O mode.

Note: *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. Once the DWIO bit has been set to 1, only H_RESET can clear it to 0. The DWIO mode setting is unaffected by S_RESET or setting of the STOP bit. Table 19 shows how the 32 bytes of address space are used in DWord I/O mode.*

All I/O resources must be accessed in DWord quantities and on DWord addresses. A read access other than listed in Table 20 will yield undefined data, a write operation may cause unexpected reprogramming of the Am79C972 control registers.

Table 18. Legal I/O Accesses in Word I/O Mode (DWIO = 0)

AD[4:0]	BE[3:0]	Type	Comment
0XX00	1110	RD	Byte read of APROM location 0h, 4h, 8h or Ch
0XX01	1101	RD	Byte read of APROM location 1h, 5h, 9h or Dh
0XX10	1011	RD	Byte read of APROM location 2h, 6h, Ah or Eh
0XX11	0111	RD	Byte read of APROM location 3h, 7h, Bh or Fh
0XX00	1100	RD	Word read of APROM locations 1h (MSB) and 0h (LSB), 5h and 4h, 8h and 9h or Ch and Dh
0XX10	0011	RD	Word read of APROM locations 3h (MSB) and 2h (LSB), 7h and 6h, Bh and Ah or Fh and Eh
10000	1100	RD	Word read of RDP
10010	0011	RD	Word read of RAP
10100	1100	RD	Word read of Reset Register
10110	0011	RD	Word read of BDP
0XX00	1100	WR	Word write to APROM locations 1h (MSB) and 0h (LSB), 5h and 4h, 8h and 9h or Ch and Dh
0XX10	0011	WR	Word write to APROM locations 3h (MSB) and 2h (LSB), 7h and 6h, Bh and Ah or Fh and Eh
10000	1100	WR	Word write to RDP
10010	0011	WR	Word write to RAP
10100	1100	WR	Word write to Reset Register
10110	0011	WR	Word write to BDP
10000	0000	WR	DWord write to RDP, switches device to DWord I/O mode

Table 19. I/O Map In DWord I/O Mode (DWIO = 1)

Offset	No. of Bytes	Register
00h - 0Fh	16	APROM
10h	4	RDP
14h	4	RAP (shared by RDP and BDP)
18h	4	Reset Register
1Ch	4	BDP

Table 20. Legal I/O Accesses in Double Word I/O Mode (DWIO =1)

AD[4:0]	BE[3:0]	Type	Comment
0XX00	0000	RD	DWord read of APROM locations 3h (MSB) to 0h (LSB), 7h to 4h, Bh to 8h or Fh to Ch
10000	0000	RD	DWord read of RDP
10100	0000	RD	DWord read of RAP
11000	0000	RD	DWord read of Reset Register
0XX00	0000	WR	DWord write to APROM locations 3h (MSB) to 0h (LSB), 7h to 4h, Bh to 8h or Fh to Ch
10000	0000	WR	DWord write to RDP
10100	0000	WR	DWord write to RAP
11000	0000	WR	DWord write to Reset Register

USER ACCESSIBLE REGISTERS

The Am79C972 controller has three types of user registers: the PCI configuration registers, the Control and Status registers (CSR), and the Bus Control registers (BCR).

The Am79C972 controller implements all PCnet-ISA (Am79C960) registers, all C-LANCE (Am79C90) registers, plus a number of additional registers. The Am79C972 CSRs are compatible upon power up with both the PCnet-ISA CSRs and all of the C-LANCE CSRs.

The PCI configuration registers can be accessed in any data width. All other registers must be accessed according to the I/O 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. 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.

The Am79C972 registers can be divided into four groups: PCI Configuration, Setup, Running, and Test. Registers not included in any of these categories can be assumed to be intended for diagnostic purposes.

■ PCI Configuration Registers

These registers are intended to be initialized by the system initialization procedure (e.g., BIOS device initialization routine) to program the operation of the Am79C972 controller PCI bus interface.

The following is a list of the registers that would typically need to be programmed once during the initialization of the Am79C972 controller within a system:

- PCI I/O Base Address or Memory Mapped I/O Base Address register
- PCI Expansion ROM Base Address register
- PCI Interrupt Line register
- PCI Latency Timer register
- PCI Status register
- PCI Command register
- OnNow register

■ Setup Registers

These registers are intended to be initialized by the device driver to program the operation of various Am79C972 controller features.

The following is a list of the registers that would typically need to be programmed once during the setup of the Am79C972 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.

Note: *Registers marked with "^" 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. Registers marked with "*" will be initialized by the initialization block read operation.*

CSR1	Initialization Block Address[15:0]
CSR2*	Initialization Block Address[31:16]
CSR3	Interrupt Masks and Deferral Control
CSR4	Test and Features Control
CSR5	Extended Control and Interrupt
CSR7	Extended Control and Interrupt2
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[15:0]
CSR13*^	Physical Address[31:16]
CSR14*^	Physical Address[47:32]
CSR15*	Mode
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*	Transmit Polling Interval
CSR49*	Receive 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
CSR116^	OnNow Miscellaneous
CSR122	Receiver Packet Alignment Control

CSR125^	MAC Enhanced Configuration Control
BCR2^	Miscellaneous Configuration
BCR4^	LED0 Status
BCR5^	LED1 Status
BCR6^	LED2 Status
BCR7^	LED3 Status
BCR9^	Full-Duplex Control
BCR18^	Bus and Burst Control
BCR19	EEPROM Control and Status
BCR20	Software Style
BCR22^	PCI Latency
BCR23^	PCI Subsystem Vendor ID
BCR24^	PCI Subsystem ID
BCR25^	SRAM Size
BCR26^	SRAM Boundary
BCR27^	SRAM Interface Control
BCR32^	MII Control and Status
BCR33^	MII Address
BCR35^	PCI Vendor ID
BCR36	PCI Power Management Capabilities (PMC) Alias Register
BCR37	PCI DATA Register Zero (DATA0) Alias Register
BCR38	PCI DATA Register One (DATA1) Alias Register
BCR39	PCI DATA Register Two (DATA2) Alias Register
BCR40	PCI DATA Register Three (DATA3) Alias Register
BCR41	PCI DATA Register Four (DATA4) Alias Register
BCR42	PCI DATA Register Five (DATA5) Alias Register
BCR43	PCI DATA Register Six (DATA6) Alias Register
BCR44	PCI DATA Register Seven (DATA7) Alias Register
BCR45	OnNow Pattern Matching Register 1
BCR46	OnNow Pattern Matching Register 2
BCR47	OnNow Pattern Matching Register 3

- **Running Registers**

These registers are intended to be used by the device driver software after the Am79C972 controller is running to access status information and to pass control information.

The following is a list of the registers that would typically need to be periodically read and perhaps written during the normal running operation of the Am79C972 controller within a system. Each of these registers contains control bits, or status bits, or both.

RAP	Register Address Port
CSR0	Am79C972 Controller Status
CSR3	Interrupt Masks and Deferral Control
CSR4	Test and Features Control
CSR5	Extended Control and Interrupt
CSR7	Extended Control and Interrupt2
CSR112	Missed Frame Count
CSR114	Receive Collision Count
BCR32	MII Control and Status
BCR33	MII Address
BCR34	MII Management Data

- **Test Registers**

These registers are intended to be used only for testing and diagnostic purposes. Those registers not included in any of the above lists can be assumed to be intended for diagnostic purposes.

PCI Configuration Registers

PCI Vendor ID Register

Offset 00h

The PCI Vendor ID register is a 16-bit register that identifies the manufacturer of the Am79C972 controller. AMD's 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 PCI Vendor ID register is located at offset 00h in the PCI Configuration Space. It is read only.

This register is the same as BCR35 and can be written by the EEPROM.

PCI Device ID Register

Offset 02h

The PCI Device ID register is a 16-bit register that uniquely identifies the Am79C972 controller within AMD's product line. The Am79C972 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 AMD. The Device ID is the same as the

PCnet-PCI II (Am79C970A) and PCnet-*FAST* (Am79C971) devices.

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

PCI Command Register

Offset 04h

The PCI Command register is a 16-bit register used to control the gross functionality of the Am79C972 controller. It controls the Am79C972 controller's ability to generate and respond to PCI bus cycles. To logically disconnect the Am79C972 device from all PCI bus cycles except configuration cycles, a value of 0 should be written to this register.

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

Bit	Name	Description
15-10	RES	Reserved locations. Read as zeros; write operations have no effect.
9	FBTBEN	Fast Back-to-Back Enable. Read as zero; write operations have no effect. The Am79C972 controller will not generate Fast Back-to-Back cycles.
8	SERREN	SERR Enable. Controls the assertion of the $\overline{\text{SERR}}$ pin. $\overline{\text{SERR}}$ is disabled when SERREN is cleared. $\overline{\text{SERR}}$ 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 by setting the STOP bit.
7	RES	Reserved location. Read as zeros; write operations have no effect.
6	PERREN	Parity Error Response Enable. Enables the parity error response functions. When PERREN is 0 and the Am79C972 controller detects a parity error, it only sets the Detected Parity Error bit in the PCI Status register. When PERREN is 1, the Am79C972 controller asserts $\overline{\text{PERR}}$ on the detection of a data parity error. It also sets the DATAPERR bit (PCI Status register, bit 8), when the data parity error occurred during a master cycle. PERREN also enables reporting address parity errors through the $\overline{\text{SERR}}$ pin and the $\overline{\text{SERR}}$ bit in the PCI Status register. PERREN is cleared by H_RESET and is not affected by S_RESET or by setting the STOP bit.
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 Am79C972 controller only generates Memory Write cycles.
3	SCYCEN	Special Cycle Enable. Read as zero; write operations have no effect. The Am79C972 controller ignores all Special Cycle operations.
2	BMEN	Bus Master Enable. Setting BMEN enables the Am79C972 controller to become a bus master on the PCI bus. The host must set BMEN before setting the INIT or STRT bit in CSR0 of the Am79C972 controller. BMEN is cleared by H_RESET and is not effected by S_RESET or by setting the STOP bit.
1	MEMEN	Memory Space Access Enable. The Am79C972 controller will ignore all memory accesses when MEMEN is cleared. The host must set MEMEN before the first memory access to the device. For memory mapped I/O, the host must program the PCI Memory Mapped I/O Base Address register with a valid memory address before setting MEMEN. For accesses to the Expansion ROM, the host must program the PCI Expansion ROM Base Address register at offset 30h with a

valid memory address before setting MEMEN. The Am79C972 controller will only respond to accesses to the Expansion ROM when both ROMEN (PCI Expansion ROM Base Address register, bit 0) and MEMEN are set to 1. Since MEMEN also enables the memory mapped access to the Am79C972 I/O resources, the PCI Memory Mapped I/O Base Address register must be programmed with an address so that the device does not claim cycles not intended for it.

MEMEN is cleared by H_RESET and is not effected by S_RESET or by setting the STOP bit.

0 IOEN — I/O Space Access Enable. The Am79C972 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 PCI I/O Base Address register 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 by setting the STOP bit.

PCI Status Register

Offset 06h

The PCI 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.

Bit	Name	Description
15	PERR	Parity Error. PERR is set when the Am79C972 controller detects a parity error. The Am79C972 controller samples the AD[31:0], 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, for all I/O, memory and configuration write commands that select the Am79C972 controller when data is transferred ($\overline{TRDY}$ and $\overline{IRDY}$ are asserted). • In master mode, during the data phase of all memory read commands. In master mode, during the data phase of the memory write command, the Am79C972 controller sets the PERR bit if the target reports a data parity error by asserting the $\overline{PERR}$ signal. PERR is not effected by the state of the Parity Error Response enable bit (PCI Command register, bit 6). PERR is set by the Am79C972 controller and cleared by writing a 1. Writing a 0 has no effect. PERR is cleared by H_RESET and is not affected by S_RESET or by setting the STOP bit.
14	SERR	Signaled SERR. SERR is set when the Am79C972 controller detects an address parity error and both SERREN and PERREN (PCI Command register, bits 8 and 6) are set. SERR is set by the Am79C972 controller and cleared by writing a 1. Writing a 0 has no effect. SERR is cleared by H_RESET and is not affected by S_RESET or by setting the STOP bit.
13	RMABORT	Received Master Abort. RMABORT is set when the Am79C972 controller terminates a master cycle with a master abort sequence. RMABORT is set by the Am79C972 controller and cleared by writing a 1. Writing a 0 has no effect. RMABORT is cleared by H_RESET and is not affected by S_RESET or by setting the STOP bit.
12	RTABORT	Received Target Abort. RTABORT is set when a target terminates an Am79C972 master cycle with a target abort sequence.

RTABORT is set by the Am79C972 controller and cleared by writing a 1. Writing a 0 has no effect. RTABORT is cleared by H_RESET and is not affected by S_RESET or by setting the STOP bit.

11 STABORT Send Target Abort. Read as zero; write operations have no effect. The Am79C972 controller will never terminate a slave access with a target abort sequence.

STABORT is read only.

10-9 DEVSEL Device Select Timing. DEVSEL is set to 01b (medium), which means that the Am79C972 controller will assert $\overline{\text{DEVSEL}}$ two clock periods after $\overline{\text{FRAME}}$ is asserted.

DEVSEL is read only.

8 DATAPERR Data Parity Error Detected. DATAPERR is set when the Am79C972 controller is the current bus master and it detects a data parity error and the Parity Error Response enable bit (PCI Command register, bit 6) is set.

During the data phase of all memory read commands, the Am79C972 controller checks for parity error by sampling the AD[31:0] and C/$\overline{\text{BE}}$[3:0] and the PAR lines. During the data phase of all memory write commands, the Am79C972 controller checks the $\overline{\text{PERR}}$ input to detect whether the target has reported a parity error.

DATAPERR is set by the Am79C972 controller and cleared by writing a 1. Writing a 0 has no effect. DATAPERR is cleared by H_RESET and is not affected by S_RESET or by setting the STOP bit.

7 FBTBC Fast Back-To-Back Capable. Read as one; write operations have no effect. The Am79C972 controller is capable of accepting fast back-to-back transactions with the first transaction addressing a different target.

6-5 RES Reserved locations. Read as zero; write operations have no effect.

4 NEW_CAP New Capabilities. This bit indicates whether this function implements a list of extended capabilities such as PCI power management. When set, this bit indicates the presence of New Capabilities. A value of 0 means that this function does not implement New Capabilities.

Read as one; write operations have no effect. The Am79C972 controller supports the Linked Additional Capabilities List.

3-0 RES Reserved locations. Read as zero; write operations have no effect.

PCI Revision ID Register

Offset 08h

The PCI Revision ID register is an 8-bit register that specifies the Am79C972 controller revision number. The value of this register is 3Xh with the lower four bits being silicon-revision dependent.

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

PCI Programming Interface Register

Offset 09h

The PCI Programming Interface register is an 8-bit register that identifies the programming interface of Am79C972 controller. PCI does not define any specific register-level programming interfaces for network devices. The value of this register is 00h.

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

PCI Sub-Class Register

Offset 0Ah

The PCI Sub-Class register is an 8-bit register that identifies specifically the function of the Am79C972 controller. The value of this register is 00h which identifies the Am79C972 device as an Ethernet controller.

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

PCI Base-Class Register

Offset 0Bh

The PCI Base-Class register is an 8-bit register that broadly classifies the function of the Am79C972 controller. The value of this register is 02h which classifies the Am79C972 device as a network controller.

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

PCI Latency Timer Register

Offset 0Dh

The PCI Latency Timer register is an 8-bit register that specifies the minimum guaranteed time the Am79C972 controller will control the bus once it starts its bus mastership period. The time is measured in clock cycles. Every time the Am79C972 controller asserts $\overline{\text{FRAME}}$ at the beginning of a bus mastership period, it will copy the value of the PCI Latency Timer register into a counter and start counting down. The counter will freeze at 0. When the system arbiter removes $\overline{\text{GNT}}$ while the counter is non-zero, the Am79C972 controller will continue with its data transfers. It will only release the bus when the counter has reached 0.

The PCI Latency Timer is only significant in burst transactions, where $\overline{\text{FRAME}}$ stays asserted until the last data phase. In a non-burst transaction, $\overline{\text{FRAME}}$ is only asserted during the address phase. The internal latency counter will be cleared and suspended while $\overline{\text{FRAME}}$ is deasserted.

All eight bits of the PCI Latency Timer register are programmable. The host should read the Am79C972 PCI MIN_GNT and PCI MAX_LAT registers to determine the latency requirements for the device and then initialize the Latency Timer register with an appropriate value.

The PCI Latency Timer register is located at offset 0Dh in the PCI Configuration Space. It is read and written by the host. The PCI Latency Timer register is cleared by H_RESET and is not effected by S_RESET or by setting the STOP bit.

PCI Header Type Register

Offset 0Eh

The PCI 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 PCI Header Type register is located at address 0Eh in the PCI Configuration Space. It is read only.

Bit	Name	Description
7	FUNCT	Single-function/multi-function device. Read as zero; write operations have no effect. The Am79C972 controller is a single function device.
6-0	LAYOUT	PCI configuration space layout. Read as zeros; write operations have no effect. The layout of the PCI configuration space locations 10h to 3Ch is as shown in the table at the beginning of this section.

PCI I/O Base Address Register

Offset 10h

The PCI I/O Base Address register is a 32-bit register that determines the location of the Am79C972 I/O resources in all of I/O space. It is located at offset 10h in the PCI Configuration Space.

Bit	Name	Description
31-5	IOBASE	I/O base address most significant 27 bits. These bits are written by the host to specify the location of the Am79C972 I/O resources in all of I/O space. IOBASE must be written with a valid address before the Am79C972 controller slave I/O mode is turned on by setting the IOEN bit (PCI Command register, bit 0). When the Am79C972 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[31:5] during the address phase of the cycles matches the value of IOBASE, the Am79C972 controller will drive $\overline{\text{DEVSEL}}$ indicating it will respond to the access. IOBASE is read and written by the host. IOBASE is cleared by H_RESET and is not affected by S_RESET or by setting the STOP bit.
4-2	IOSIZE	I/O size requirements. Read as zeros; write operations have no effect. IOSIZE indicates the size of the I/O space the Am79C972 controller requires. When the host writes a value of FFFF FFFFh to the I/O Base Address register, it will read back a value of 0 in bits 4-2. That indicates an Am79C972 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.

PCI Memory Mapped I/O Base Address Register

Offset 14h

The PCI Memory Mapped I/O Base Address register is a 32-bit register that determines the location of the Am79C972 I/O resources in all of memory space. It is located at offset 14h in the PCI Configuration Space.

Bit	Name	Description
31-5	MEMBASE	Memory mapped I/O base address most significant 27 bits. These bits are written by the host to specify the location of the Am79C972 I/O resources in all of memory space. MEMBASE must be written with a valid address before the Am79C972 controller slave memory mapped I/O mode is turned on by setting the MEMEN bit (PCI Command register, bit 1). When the Am79C972 controller is enabled for memory mapped I/O mode (MEMEN is set), it monitors the PCI bus for a valid memory command. If the value on AD[31:5] during the address phase of the cycles matches the value of MEMBASE, the Am79C972 controller will drive $\overline{\text{DEVSEL}}$ indicating it will respond to the access. MEMBASE is read and written by the host. MEMBASE is cleared by H_RESET and is not affected by S_RESET or by setting the STOP bit.
4	MEMSIZE	Memory mapped I/O size requirements. Read as zeros; write operations have no effect. MEMSIZE indicates the size of the memory space the Am79C972 controller requires. When the host writes a value of FFFF FFFFh to the Memory Mapped I/O Base Address register, it will read back a value of 0 in bit 4. That indicates a Am79C972 memory space requirement of 32 bytes.
3	PREFETCH	Prefetchable. Read as zero; write operations have no effect. Indicates that memory space controlled by this base address register is not prefetchable. Data in the memory mapped I/O space cannot be prefetched. Because one of the I/O resources in this address space is a Reset register, the order of the read accesses is important.
2-1	TYPE	Memory type indicator. Read as zeros; write operations have no effect. Indicates that this base address register is 32 bits wide and mapping can be done anywhere in the 32-bit memory space.
0	MEMSPACE	Memory space indicator. Read as zero; write operations have no effect. Indicates that this base address register describes a memory base address.

PCI Subsystem Vendor ID Register

Offset 2Ch

The PCI Subsystem Vendor ID register is a 16-bit register that together with the PCI Subsystem ID uniquely identifies the add-in card or subsystem the Am79C972 controller is used in. Subsystem Vendor IDs can be obtained from the PCI SIG. A value of 0 (the default) indicates that the Am79C972 controller does not support subsystem identification. The PCI Subsystem Vendor ID is an alias of BCR23, bits 15-0. It is programmable through the EEPROM.

The PCI Subsystem Vendor ID register is located at offset 2Ch in the PCI Configuration Space. It is read only.

PCI Subsystem ID Register

Offset 2Eh

The PCI Subsystem ID register is a 16-bit register that together with the PCI Subsystem Vendor ID uniquely identifies the add-in card or subsystem the Am79C972 controller is used in. The value of the Subsystem ID is up to the system vendor. A value of 0 (the default) indicates that the Am79C972 controller does not support subsystem identification. The PCI Subsystem ID is an alias of BCR24, bits 15-0. It is programmable through the EEPROM.

The PCI Subsystem ID register is located at offset 2Eh in the PCI Configuration Space. It is read only.

PCI Expansion ROM Base Address Register

Offset 30h

The PCI Expansion ROM Base Address register is a 32-bit register that defines the base address, size and address alignment of an Expansion ROM. It is located at offset 30h in the PCI Configuration Space.

Bit	Name	Description
31-20	ROMBASE	Expansion ROM base address most significant 12 bits. These bits are written by the host to specify the location of the Expansion ROM in all of memory space. ROMBASE must be written with a valid address before the Am79C972 Expansion ROM access is enabled by setting ROMEN (PCI Expansion ROM Base Address register, bit 0) and MEMEN (PCI Command register, bit 1). Since the 12 most significant bits of the base address are programmable, the host can map the Expansion ROM on any 1M boundary. When the Am79C972 controller is enabled for Expansion ROM access (ROMEN and MEMEN are set to 1), it monitors the PCI bus for a valid memory command. If the value on AD[31:2] during the address phase of the cycle falls between ROMBASE and ROMBASE + 1M - 4, the Am79C972 controller will drive $\overline{\text{DEVSEL}}$ indicating it will respond to the access. ROMBASE is read and written by the host. ROMBASE is cleared by H_RESET and is not affected by S_RESET or by setting the STOP bit.
19-1	ROMSIZE	ROM size. Read as zeros; write operation have no effect. ROMSIZE indicates the maximum size of the Expansion ROM the Am79C972 controller can support. The host can determine the Expansion ROM size by writing FFFF FFFFh to the Expansion ROM Base Address register. It will read back a value of 0 in bit 19-1, indicating an Expansion ROM size of 1M. Note that ROMSIZE only specifies the maximum size of Expansion ROM the Am79C972 controller supports. A smaller ROM can be used, too. The actual size of the code in the Expansion ROM is always determined by reading the Expansion ROM header.
0	ROMEN	Expansion ROM Enable. Written by the host to enable access to the Expansion ROM. The Am79C972 controller will only respond to accesses to the Expansion ROM when both ROMEN and MEMEN (PCI Command register, bit 1) are set to 1. ROMEN is read and written by the host. ROMEN is cleared by H_RESET and is not effected by S_RESET or by setting the STOP bit.

PCI Capabilities Pointer Register

Offset 34h

Bit	Name	Description
7-0	CAP_PTR	The PCI Capabilities pointer Register is an 8-bit register that points to a linked list of capabilities implemented on this device. This register has a default value of 40h. The PCI Capabilities register is located at offset 34h in the PCI Configuration Space. It is read only.

PCI Interrupt Line Register

Offset 3Ch

The PCI 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 initializes the Am79C972 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 Am79C972 controller. The PCI Interrupt Line register is not modified by the Am79C972 controller. It has no effect on the operation of the device.

The PCI Interrupt Line register is located at offset 3Ch in the PCI Configuration Space. It is read and written by

the host. It is cleared by H_RESET and is not affected by S_RESET or by setting the STOP bit.

PCI Interrupt Pin Register

Offset 3Dh

This PCI Interrupt Pin register is an 8-bit register that indicates the interrupt pin that the Am79C972 controller is using. The value for the Am79C972 Interrupt Pin register is 01h, which corresponds to $\overline{\text{INTA}}$.

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

PCI MIN_GNT Register

Offset 3Eh

The PCI MIN_GNT register is an 8-bit register that specifies the minimum length of a burst period that the Am79C972 device needs to keep up with the network activity. The length of the burst period is calculated assuming a clock rate of 33 MHz. The register value specifies the time in units of 1/4 µs. The PCI MIN_GNT register is an alias of BCR22, bits 7-0. It is recommended that the BCR22 be programmed to a value of 1818h.

The host should use the value in this register to determine the setting of the PCI Latency Timer register.

The PCI MIN_GNT register is located at offset 3Eh in the PCI Configuration Space. It is read only.

PCI MAX_LAT Register

Offset 3Fh

The PCI MAX_LAT register is an 8-bit register that specifies the maximum arbitration latency the Am79C972 controller can sustain without causing problems to the network activity. The register value specifies the time in units of 1/4 µs. The MAX_LAT register is an alias of BCR22, bits 15-8. It is recommended that BCR22 be programmed to a value of 1818h.

The host should use the value in this register to determine the setting of the PCI Latency Timer register.

The PCI MAX_LAT register is located at offset 3Fh in the PCI Configuration Space. It is read only

PCI Capability Identifier Register

Offset 40h

Bit	Name	Description
7-0	CAP_ID	This register, when set to 1, identifies the linked list item as being the PCI Power Management registers. This register has a default value of 1h. The PCI Capabilities Identifier register is located at offset 40h in the PCI Configuration Space. It is read only.

PCI Next Item Pointer Register

Offset 41h

Bit	Name	Description
7-0	NXT_ITM_PTR	The Next Item Pointer Register points to the starting address of the next capability. The pointer at this offset is a null pointer, indicating that this is the last capability in the linked list of the capabilities. This register has a default value of 0h. The PCI Next Pointer Register is located at offset 41h in the PCI Configuration Space. It is read only.

PCI Power Management Capabilities Register (PMC)

Offset 42h

Note: *All bits of this register are loaded from EEPROM. The register is aliased to BCR36 for testing purposes.*

Bit	Name	Description
15-11	PME_SPT	PME Support. This 5-bit field indicates the power states in which the function may assert $\overline{\text{PME}}$. A value of 0b for any bit indicates that the function is not capable of asserting the $\overline{\text{PME}}$ signal while in that power state. Bit(11) XXXX1b - $\overline{\text{PME}}$ can be asserted from D0. Bit(12) XXX1Xb - $\overline{\text{PME}}$ can be asserted from D1. Bit(13) XX1XXb - $\overline{\text{PME}}$ can be asserted from D2. Bit(14) X1XXXb - $\overline{\text{PME}}$ can be asserted from D3hot. Bit(15) 1XXXXb - $\overline{\text{PME}}$ can be asserted from D3cold. Read only.

10	D2_SPT	D2 Support. If this bit is a 1, this function supports the D2 Power Management State. Read only.
9	D1_SPT	D1 Support. If this bit is a 1, this function supports the D1 Power Management State. Read only.
8-6	RES	Reserved locations. Written as zeros and read as undefined.
5	DSI	Device Specific Initialization. When this bit is 1, it indicates that special initialization of the function is required (beyond the standard PCI configuration header) before the generic class device driver is able to use it. Read only.
4	AUXPS	Auxiliary Power Source. This bit is only meaningful if bit 15 (D3cold supporting $\overline{\text{PME}}$) is a 1. When this bit is also a 1, it indicates that support for $\overline{\text{PME}}$ in D3cold requires auxiliary power supplied by the system by way of a proprietary delivery vehicle. A 0 in this bit indicates that the function supplies its own auxiliary power source. If the function does not support $\overline{\text{PME}}$ while in D3cold, (bit15=0) then this field must always return 0. Read only.
3	PME_CLK	PME Clock. When this bit is a 1, it indicates that the function relies on the presence of the PCI clock for $\overline{\text{PME}}$ operation. When this bit is a 0 it indicates that no PCI clock is required for the function to generate $\overline{\text{PME}}$. Functions that do not support $\overline{\text{PME}}$ generation in any state must return 0 for this field. Read only.
2-0	PMIS_VER	Power Management Interface Specification Version. A value of 001b indicates that this function complies with the revision 1.0 of the PCI Power Management Interface Specification.

PCI Power Management Control/Status Register (PMCSR)

Offset 44h

Bit	Name	Description
15	PME_STATUS	PME Status. This bit is set when the function would normally assert the $\overline{\text{PME}}$ signal independent of the state of the PME_EN bit. Writing a 1 to this bit will clear it and cause the function to stop asserting a $\overline{\text{PME}}$ (if enabled). Writing a 0 has no effect. If the function supports $\overline{\text{PME}}$ from D3cold then this bit is sticky and must be explicitly cleared by the operating system each time the operating system is initially loaded. Read/write accessible always. Sticky bit. This bit is reset by POR. H_RESET, S_RESET, or setting the STOP bit has no effect.
14-13	DATA_SCALE	Data Scale. This two bit read-only field indicates the scaling factor to be used when interpreting the value of the Data register. The value and meaning of this field will vary depending on the DATA_SCALE field. Read only.
12-9	DATA_SEL	Data Select. This optional four-bit field is used to select which data is reported through the Data register and DATA_SCALE field. Read/write accessible always. Sticky bit. This bit is reset by POR. H_RESET, S_RESET, or setting the STOP bit has no effect.

8	PME_EN	PME Enable. When a 1, PME_EN enables the function to assert $\overline{\text{PME}}$. When a 0, $\overline{\text{PME}}$ assertion is disabled. This bit defaults to "0" if the function does not support $\overline{\text{PME}}$ generation from D3cold. If the function supports $\overline{\text{PME}}$ from D3cold, then this bit is sticky and must be explicitly cleared by the operating system each time the operating system is initially loaded. Read/write accessible always. Sticky bit. This bit is reset by POR. H_RESET, S_RESET, or setting the STOP bit has no effect.
7-2	RES	Reserved locations. Read only.
1-0	PWR_STATE	Power State. This 2-bit field is used both to determine the current power state of a function and to set the function into a new power state. The definition of the field values is given below. 00b - D0. 01b - D1. 10b - D2. 11b - D3. These bits can be written and read, but their contents have no effect on the operation of the device. Read/write accessible always.

PCI PMCSR Bridge Support Extensions Register

Offset 46h

Bit	Name	Description
7-0	PMCSR_BSE	The PCI PMCSR Bridge Support Extensions Register is an 8-bit register. PMCSR Bridge Support Extensions are not supported. This register has a default value of 00h. The PCI PMCSR Bridge Support Extensions register is located at offset 46h in the PCI Configuration Space. It is read only.

PCI Data Register

Offset 47h

Note: *All bits of this register are loaded from EEPROM. The register is aliased to lower bytes of the BCR37-44 for testing purposes.*

Bit	Name	Description
7-0	DATA_REG	The PCI Data Register is an 8-bit register. Refer to the "PCI Bus Power Management Interface Specification" version 1.0 for a more detailed description of this register. The PCI DATA register is located at offset 47h 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 Am79C972 controller. The RAP contains the address of a CSR or BCR.

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, 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

Am79C972 control registers. A read access will yield undefined values.

Read/Write accessible always. RAP is cleared by H_RESET or S_RESET and is unaffected by setting 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.

CSR0: Am79C972 Controller Status and Control Register

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.

Bit	Name	Description
31-16	RES	Reserved locations. Written as zeros and read as undefined.
15	ERR	Error is set by the OR of CERR, MISS, and MERR. ERR remains set as long as any of the error flags are true. Read accessible always. ERR is read only. Write operations are ignored.
14	RES	Reserved locations. Read/Write accessible always. Read returns zero.
13	CERR	Collision Error is set by the Am79C972 controller when the device operates in half-duplex mode and the collision inputs to the GPSI port failed to activate within 20 network bit times after the chip terminated transmission (SQE Test). This feature is a transceiver test feature. CERR reporting is disabled when the GPSI port is active and the Am79C972 controller operates in full-duplex mode. When the MII port is selected, CERR is only reported when the external PHY is operating as a half-duplex 10BASE-T PHY. CERR assertion will not result in an interrupt being generated. CERR assertion will set the ERR bit. Read/Write accessible always. CERR is cleared by the host by writing a 1. Writing a 0 has no effect. CERR is cleared by H_RESET, S_RESET, or by setting the STOP bit.
12	MISS	Missed Frame is set by the Am79C972 controller when it 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. The Missed Frame Counter (CSR112) also increments each time a receive frame is missed. When MISS is set, $\overline{\text{INTA}}$ is asserted if IENA is 1 and the mask bit MISSM (CSR3, bit 12) is 0. MISS assertion will set the ERR bit, regardless of the settings of IENA and MISSM. Read/Write accessible always. MISS is cleared by the host by writing a 1. Writing a 0 has no effect. MISS is cleared by H_RESET, S_RESET, or by setting the STOP bit.
11	MERR	Memory Error is set by the Am79C972 controller when it requests the use of the system interface bus by asserting $\overline{\text{REQ}}$ and has not received $\overline{\text{GNT}}$ 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 153.6 µs of bus latency.

When MERR is set, $\overline{INTA}$ is asserted if IENA is 1 and the mask bit MERRM (CSR3, bit 11) is 0. MERR assertion will set the ERR bit, regardless of the settings of IENA and MERRM.

Read/Write accessible always. MERR is cleared by the host 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 is set by the Am79C972 controller after the last descriptor of a receive frame has been updated by writing a 0 to the OWNership bit. RINT may also be set when the first descriptor of a receive frame has been updated by writing a 0 to the OWNership bit if the LAPPEN bit of CSR3 has been set to a 1.

When RINT is set, $\overline{INTA}$ is asserted if IENA is 1 and the mask bit RINTM (CSR3, bit 10) is 0.

Read/Write accessible always. 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 by the Am79C972 controller 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 is 1 and the mask bit TINTM (CSR3, bit 9) is 0.

TINT will not be set if TINTOKD (CSR5, bit 15) is set to 1 and the transmission was successful.

Read/Write accessible always. TINT is cleared by the host by writing a 1. Writing a 0 has no effect. TINT is cleared by H_RESET, S_RESET, or by setting the STOP bit.

8 IDON — Initialization Done is set by the Am79C972 controller after the initialization sequence has completed. When IDON is set, the Am79C972 controller has read the initialization block from memory.

When IDON is set, $\overline{INTA}$ is asserted if IENA is 1 and the mask bit IDONM (CSR3, bit 8) is 0.

Read/Write accessible always. IDON is cleared by the host by writing a 1. Writing a 0 has no effect. IDON is cleared by H_RESET, S_RESET, or by setting the STOP bit.

7 INTR — Interrupt Flag indicates that one or more following interrupt causing conditions has occurred: EXDINT, IDON, MERR, MISS, MFCO, RCVCCO, RINT, SINT, TINT, TXSTRT, UINT, STINT, MREINT, MCCINT, MIIPDTINT, MAPINT and the associated mask or enable bit is programmed to allow the event to cause an interrupt. If IENA is set to 1 and INTR is set, $\overline{INTA}$ will be active. When INTR is set by SINT or SLPINT, $\overline{INTA}$ will be active independent of the state of IENA.

Read accessible always. INTR is read only. INTR is cleared by clearing all of the active individual interrupt bits that have not been masked out.

6 IENA — Interrupt Enable allows $\overline{INTA}$ to be active if the Interrupt Flag is set. If IENA = 0, then $\overline{INTA}$ will be disabled regardless of the state of INTR.

Read/Write accessible always. IENA is set by writing a 1 and cleared by writing a 0. IENA is cleared by H_RESET or S_RESET and setting the STOP bit.

5 RXON — Receive On indicates that the receive function is enabled. RXON is set if DRX (CSR15, bit 0) is set to 0 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.

Read accessible always. RXON is read only. RXON is cleared by H_RESET or S_RESET and setting the STOP bit.

4 TXON — Transmit On indicates that the transmit function is enabled. TXON is set if DTX (CSR15, bit 1) is set to 0 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.

This bit will reset if the DXSUFLO bit (CSR3, bit 6) is reset and there is an underflow condition encountered.

Read accessible always. TXON is read only. TXON is cleared by H_RESET or S_RESET and 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 TXDPOLL bit in CSR4 is set. Setting TDMD while TXDPOLL = 0 merely hastens the Am79C972 controller's response to a Transmit Descriptor Ring Entry.

Read/Write accessible always. 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.

Read/Write accessible always. 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 Am79C972 controller to send and receive frames, and perform buffer management operations. Setting STRT clears the STOP bit. If STRT and INIT are set together, the Am79C972 controller initialization will be performed first.

Read/Write accessible always. 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 the Am79C972 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, the Am79C972 controller initialization will be performed first. INIT is not cleared when the initialization sequence has completed.

Read/Write accessible always. 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: Initialization Block Address 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 0 to align the initialization block to a DWord boundary. This register is aliased with CSR16.

Read/Write accessible only when either the STOP or the SPND bit is set. Unaffected by H_RESET or S_RESET, or by setting the STOP bit.

CSR2: Initialization Block Address 1

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 the Am79C972 bus master accesses, while the 32-bit hardware for which the Am79C972 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 Am79C972 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 either the STOP or the SPND bit 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 either the STOP or the SPND bit 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.
15-13	RES	Reserved locations. Read and written as zero.
12	MISSM	Missed Frame Mask. If MISSM is set, the MISS bit will be masked and unable to set the INTR bit. 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 will be masked and unable to set the INTR bit. 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 will be masked and unable to set the INTR bit. 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 will be masked and unable to set the INTR bit.

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 will be masked and unable to set the INTR bit.

Read/Write accessible always. IDONM is cleared by H_RESET or S_RESET and is not affected by STOP.

7 RES Reserved location. Read and written as zeros.

6 DXSUFLO Disable Transmit Stop on Underflow error.

When DXSUFLO (CSR3, bit 6) is set to 0, the transmitter is turned off when an UFLO error occurs (CSR0, TXON = 0).

When DXSUFLO is set to 1, the Am79C972 controller gracefully recovers from an UFLO error. It scans the transmit descriptor ring until it finds the start of a new frame and starts a new transmission.

Read/Write accessible always. DXSUFLO is cleared by H_RESET or S_RESET and is not affected by STOP.

5 LAPPEN Look Ahead Packet Processing Enable. When set to a 1, the LAPPEN bit will cause the Am79C972 controller to generate an interrupt following the descriptor write operation to the first buffer of a receive frame. 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 1 also enables the Am79C972 controller to read the STP bit of receive descriptors. The Am79C972 controller will use the STP information to determine where it should begin writing a receive packet's data. Note that while in this mode, the Am79C972 controller can write intermediate packet data to buffers whose descriptors do not contain STP bits set to 1. Following the write to the last descriptor used by a packet, the Am79C972 controller will scan through the next descriptor entries to locate the next STP bit that is set to a 1. The Am79C972 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 = 1, then some descriptors/buffers may be skipped in the ring. While performing the search for the next STP bit that is set to 1, the Am79C972 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 Am79C972 controller ownership of the descriptor but also indicate STP = 0, then the Am79C972 controller will reset the OWN bit to 0 in these entries. If a scanned entry indicates host ownership with STP = 0, then the Am79C972 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 Am79C972 controller, then the Am79C972 controller will stop advancing through the ring en-

tries 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 Am79C972 controller, then the Am79C972 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 0 by H_RESET or S_RESET and will be unaffected by STOP.

See Appendix B for more information on the Look Ahead Packet Processing concept.

4 DXMT2PD Disable Transmit Two Part Deferral (see Medium Allocation section in the *Media Access Management* section for 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* section 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 1, big Endian mode is selected. When BSWP is set to 0, little Endian mode is selected.

When big Endian mode is selected, the Am79C972 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 Byte 0, AD[23:16] becomes Byte 1, AD[15:8] becomes Byte 2, and AD[7:0] becomes Byte 3 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 Byte 3, AD[23:16] is Byte 2, AD[15:8] is Byte 1, and AD[7:0] is Byte 0.

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, BDP and PCI configuration space accesses are not affected by the setting of the BSWP bit. Address PROM transfers are not affected by the setting of the BSWP bit. Expansion ROM accesses 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 should not be set to 1 when the Am79C972 controller is used in a PCI bus application.

Read/Write accessible always. BSWP is cleared by H_RESET or S_RESET and is not affected by STOP.

1-0 RES Reserved location. The default value of this bit is a 0. Writing a 1 to this bit has no effect on device function. If a 1 is written to this bit, then a 1 will be read back. Existing drivers may write a 1 to this bit for compatibility, but new drivers

should write a 0 to this bit and should treat the read value as undefined.

CSR4: Test and Features Control

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.

Bit	Name	Description
31-16	RES	Reserved locations. Written as zeros and read as undefined.
15	RES	Reserved location. It is OK for legacy software to write a 1 to this location. This bit must be set back to 0 before setting INIT or STRT bits. Read/Write accessible always. This bit is cleared by H_RESET or S_RESET and is unaffected by the STOP bit.
14	DMAPLUS	Writing and reading from this bit has no effect. DMAPLUS is always set to 1.
13	RES	Reserved Location. Written as zero and read as undefined.
12	TXDPOLL	Disable Transmit Polling. If TXDPOLL is set, the Buffer Management Unit will disable transmit polling. Likewise, if TXDPOLL is cleared, automatic transmit polling is enabled. If TXDPOLL 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. Transmit polling will take place following Receive activities. Read/Write accessible always. TXDPOLL 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 (CSR15, bit 3) and of the ADD_FCS bit (TMD1, bit 29) for frames shorter than 64 bytes. Read/Write accessible always. 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 always. ASTRP_RCV is cleared by H_RESET or S_RESET and is unaffected by the STOP bit.
9	MFCO	Missed Frame Counter Overflow is set by the Am79C972 controller when the Missed Frame Counter (CSR112 and CSR114) has wrapped around. When MFCO is set, $\overline{\text{INTA}}$ is asserted if IENA is 1 and the mask bit MFCOM is 0. Read/Write accessible always. MFCO is cleared by the host by writing a 1. Writing a 0 has no effect. MFCO is cleared by H_RESET, S_RESET, or by setting the STOP bit.
8	MFCOM	Missed Frame Counter Overflow Mask. If MFCOM is set, the MFCO bit will be masked and unable to set the INTR bit. Read/Write accessible always. MFCOM is set to 1 by H_RESET or S_RESET and is not affected by the STOP bit.
7	UINTCMD	User Interrupt Command. UINTCMD can be used by the host to generate an interrupt unrelated to any network activity. When UINTCMD is set, $\overline{\text{INTA}}$ is asserted if IENA is set to 1.

Writing a 1 to UNIT will clear UNITCMD and stop interrupts.

Read/Write accessible always. UINTCMD is cleared by H_RESET or S_RESET or by setting the STOP bit.

6 UINT User Interrupt. UINT is set by the Am79C972 controller after the host has issued a user interrupt command by setting UINTCMD (CSR4, bit 7) to 1.

Read/Write accessible always. UINT is cleared by the host by writing a 1. Writing a 0 has no effect. UINT is cleared by H_RESET or S_RESET or by setting the STOP bit.

5 RCVCCO Receive Collision Counter Overflow is set by the Am79C972 controller when the Receive Collision Counter (CSR114 and CSR115) has wrapped around.

When RCVCCO is set, $\overline{\text{INTA}}$ is asserted if IENA is 1 and the mask bit RCVCCOM is 0.

Read/Write accessible always. RCVCCO is cleared by the host by writing a 1. Writing a 0 has no effect. RCVCCO is cleared by H_RESET, S_RESET, or by setting the STOP bit.

4 RCVCCOM Receive Collision Counter Overflow Mask. If RCVCCOM is set, the RCVCCO bit will be masked and unable to set the INTR bit.

Read/Write accessible always. RCVCCOM is set to 1 by H_RESET or S_RESET and is not affected by the STOP bit.

3 TXSTRT Transmit Start status is set by the Am79C972 controller whenever it begins transmission of a frame.

When TXSTRT is set, $\overline{\text{INTA}}$ is asserted if IENA is 1 and the mask bit TXSTRTM is 0.

Read/Write accessible always. TXSTRT is cleared by the host by writing a 1. Writing a 0 has no effect. TXSTRT is cleared by H_RESET, S_RESET, or by setting the STOP bit.

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

Read/Write accessible always. TXSTRTM is set to 1 by H_RESET or S_RESET and is not affected by the STOP bit.

1-0 RES Reserved locations. Written as zeros and read as undefined.

CSR5: Extended Control and Interrupt 1

Certain bits in CSR5 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 CSR5 and write back the value just read to clear the interrupt condition.

Bit	Name	Description
31-16	RES	Reserved locations. Written as zeros and read as undefined.
15	TOKINTD	Transmit OK Interrupt Disable. If TOKINTD is set to 1, the TINT bit in CSR0 will not be set when a transmission was successful. Only a transmit error will set the TINT bit. TOKINTD has no effect when LTINTEN (CSR5, bit 14) is set to 1. A transmit descriptor with LTINT set to 1 will always cause TINT to be set to 1, independent of the success of the transmission. Read/Write accessible always. TOKINTD is cleared by H_RESET or S_RESET and is unaffected by STOP.
14	LTINTEN	Last Transmit Interrupt Enable. When set to 1, the LTINTEN bit will cause the Am79C972 controller to read bit 28 of TMD1 as LTINT. The setting LTINT will determine if TINT will be set at the end of the transmission.

Read/Write accessible always. LTINTEN is cleared by H_RESET or S_RESET and is unaffected by STOP.

13-12 RES — Reserved locations. Written as zeros and read as undefined.

11 SINT — System Interrupt is set by the Am79C972 controller when it detects a system error during a bus master transfer on the PCI bus. System errors are data parity error, master abort, or a target abort. The setting of SINT due to data parity error is not dependent on the setting of PERREN (PCI Command register, bit 6).

When SINT is set, $\overline{\text{INTA}}$ is asserted if the enable bit SINTE is 1. Note that the assertion of an interrupt due to SINT is not dependent on the state of the INEA bit, since INEA is cleared by the STOP reset generated by the system error.

Read/Write accessible always. SINT is cleared by the host by writing a 1. Writing a 0 has no effect. The state of SINT is not affected by clearing any of the PCI Status register bits that get set when a data parity error (DATAPERR, bit 8), master abort (RMABORT, bit 13), or target abort (RTABORT, bit 12) occurs. SINT is cleared by H_RESET or S_RESET and is not affected by setting the STOP bit.

10 SINTE — System Interrupt Enable. If SINTE is set, the SINT bit will be able to set the INTR bit.

Read/Write accessible always. SINTE is set to 0 by H_RESET or S_RESET and is not affected by setting the STOP bit.

9-8 RES — Reserved locations. Written as zeros and read as undefined.

7 EXDINT — Excessive Deferral Interrupt is set by the Am79C972 controller when the transmitter has experienced Excessive Deferral on a transmit frame, where Excessive Deferral is defined in the ISO 8802-3 (IEEE/ANSI 802.3) standard.

When EXDINT is set, $\overline{\text{INTA}}$ is asserted if the enable bit EXDINTE is 1.

Read/Write accessible always. EXDINT is cleared by the host by writing a 1. Writing a 0 has no effect. EXDINT is cleared by H_RESET and is not affected by S_RESET or setting the STOP bit.

6 EXDINTE — Excessive Deferral Interrupt Enable. If EXDINTE is set, the EXDINT bit will be able to set the INTR bit.

Read/Write accessible always. EXDINTE is set to 0 by H_RESET and is not affected by S_RESET or setting the STOP bit.

5 MPPLBA — Magic Packet Physical Logical Broadcast Accept. If MPPLBA is at its default value of 0, the Am79C972 controller will only detect a Magic Packet frame if the destination address of the packet matches the content of the physical address register (PADR). If MPPLBA is set to 1, the destination address of the Magic Packet frame can be unicast, multicast, or broadcast. Note that the setting of MPPLBA only affects the address detection of the Magic Packet frame. The Magic Packet frame's data sequence must be made up of 16 consecutive physical addresses (PADR[47:0]) regardless of what kind of destination address it has. This bit is OR'ed with EMPPLBA bit (CSR116, bit 6).

Read/Write accessible always. MPPLBA is set to 0 by H_RESET or S_RESET and is not affected by setting the STOP bit.

4 MPINT — Magic Packet Interrupt. Magic Packet Interrupt is set by the Am79C972 controller when the device is in the Magic Packet

		mode and the Am79C972 controller receives a Magic Packet frame. When MPINT is set to 1, INTA is asserted if IENA (CSR0, bit 6) and the enable bit MPINTE are set to 1. Read/Write accessible always. MPINT is cleared by the host by writing a 1. Writing a 0 has no affect. MPINT is cleared by H_RESET, S_RESET, or by setting the STOP bit.
3	MPINTE	Magic Packet Interrupt Enable. If MPINTE is set to 1, the MPINT bit will be able to set the INTR bit. Read/Write accessible always. MPINT is cleared to 0 by H_RESET or S_RESET and is not affected by setting the STOP bit.
2	MPEN	Magic Packet Enable. MPEN allows activation of the Magic Packet mode by the host. The Am79C972 controller will enter the Magic Packet mode when both MPEN and MPMODE are set to 1. Read/Write accessible always. MPEN is cleared to 0 by H_RESET or S_RESET and is not affected by setting the STOP bit.
1	MPMODE	The Am79C972 controller will enter the Magic Packet mode when MPMODE is set to 1 and either PG is asserted or MPEN is set to 1. Read/Write accessible always. MPMODE is cleared to 0 by H_RESET or S_RESET and is not affected by setting the STOP bit
0	SPND	Suspend. Setting SPND to 1 will cause the Am79C972 controller to start requesting entrance into suspend mode. The host must poll SPND until it reads back 1 to determine that the Am79C972 controller has entered the suspend mode. Setting SPND to 0 will get the Am79C972 controller out of suspend mode. SPND can only be set to 1 if STOP (CSR0, bit 2) is set to 0. H_RESET, S_RESET or setting the STOP bit will get the Am79C972 controller out of suspend mode. Requesting entrance into the suspend mode by the host depends on the setting of the FASTSPNDE bit (CSR7, bit 15). Refer to the bit description of the FASTSPNDE bit and the Suspend section in *Detailed Functions, Buffer Management Unit* for details. In suspend mode, all of the CSR and BCR registers are accessible. As long as the Am79C972 controller is not reset while in suspend mode (by H_RESET, S_RESET or by setting the STOP bit), no re-initialization of the device is required after the device comes out of suspend mode. The Am79C972 controller will continue at the transmit and receive descriptor ring locations, from where it had left, when it entered the suspend mode. Read/Write accessible always. SPND is cleared by H_RESET, S_RESET, or by setting the STOP bit.

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 the Am79C972 controller initialization. This field is written during the Am79C972 controller initialization routine. Read accessible only when either the STOP or the SPND 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 Am79C972 controller initialization. This field is written during the Am79C972 controller initialization routine.

Read accessible only when either the STOP or the SPND 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 0s. Write operations are ignored.

CSR7: Extended Control and Interrupt 2

Certain bits in CSR7 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 CSR7 and write back the value just read to clear the interrupt condition.

Bit	Name	Description
31-16	RES	Reserved locations. Written as zeros and read as undefined.
15	FASTSPNDE	Fast Suspend Enable. When FASTSPNDE is set to 1, the Am79C972 controller performs a fast suspend whenever the SPND bit is set. When a fast suspend is requested, the Am79C972 controller performs a quick entry into the suspend mode. At the time the SPND bit is set, the Am79C972 controller will complete the DMA process of any transmit and/or receive packet that had already begun DMA activity. In addition, any transmit packet that had started transmission will be fully transmitted and any receive packet that had begun reception will be fully received. However, no additional packets will be transmitted or received and no additional transmit or receive DMA activity will begin. Hence, the Am79C972 controller may enter the suspend mode with transmit and/or receive packets still in the FIFOs or the SRAM. When FASTSPNDE is 0 and the SPND bit is set, the Am79C972 controller may take longer before entering the suspend mode. At the time the SPND bit is set, the Am79C972 controller will complete the DMA process of a transmit packet if it had already begun and the Am79C972 controller will completely receive a receive packet if it had already begun. Additionally, all transmit packets stored in the transmit FIFOs and the transmit buffer area in the SRAM (if one is enabled) will be transmitted and all receive packets stored in the receive FIFOs, and the receive buffer area in the SRAM (if one is enabled) will be transferred into system memory. Since the FIFO and SRAM contents are flushed, it may take much longer before the Am79C972 controller enters the suspend mode. The amount of time that it takes depends on many factors including the size of the SRAM, bus latency, and network traffic level. When a write to CSR5 is performed with bit 0 (SPND) set to 1, the value that is simultaneously written to FASTSPNDE is used to determine which approach is used to enter suspend mode. Read/Write accessible always. FASTSPNDE is cleared by H_RESET, S_RESET or by setting the STOP bit.
14	RXFRTG	Receive Frame Tag. When Receive Frame Tag is set to 1, a tag word is put into the receive descriptor supplied by the EADI. See the section *Receive Frame Tagging* for details. This bit is valid only when the EADISEL (BCR2, bit 3) is set to 1. Read/Write accessible always. RXFRTG is cleared by H_RESET. RXFRTG is unaffect-

ed by S_RESET or by setting the STOP bit.

13 RDMD Receive Demand, when set, causes the Buffer Management Unit to access the Receive Descriptor Ring without waiting for the receive poll-time counter to elapse. If RXON is not enabled, RDMD has no meaning and no receive Descriptor Ring access will occur.

RDMD is required to be set if the RXDPOLL bit in CSR7 is set. Setting RDMD while RXDPOLL = 0 merely hastens the Am79C972 controller's response to a receive Descriptor Ring Entry.

Read/Write accessible always. RDMD is set by writing a 1. Writing a 0 has no effect. RDMD will be cleared by the Buffer Management Unit when it fetches a receive Descriptor. RDMD is cleared by H_RESET. RDMD is unaffected by S_RESET or by setting the STOP bit.

12 RXDPOLL Receive Disable Polling. If RXDPOLL is set, the Buffer Management Unit will disable receive polling. Likewise, if RXDPOLL is cleared, automatic receive polling is enabled. If RXDPOLL is set, RDMD bit in CSR7 must be set in order to initiate a manual poll of a receive descriptor. Receive Descriptor Polling will not take place if RXON is reset.

Read/Write accessible always. RXDPOLL is cleared by H_RESET. RXDPOLL is unaffected by S_RESET or by setting the STOP bit.

11 STINT Software Timer Interrupt. The Software Timer interrupt is set by the Am79C972 controller when the Software Timer counts down to 0. The Software Timer will immediately load the STVAL (BCR 31, bits 5-0) into the Software Timer and begin counting down.

When STINT is set to 1, $\overline{\text{INTA}}$ is asserted if the enable bit STINTE is set to 1.

Read/Write accessible always. STINT is cleared by the host by writing a 1. Writing a 0 has no effect. STINT is cleared by H_RESET and is not affected by S_RESET or setting the STOP bit.

10 STINTE Software Timer Interrupt Enable. If STINTE is set, the STINT bit will be able to set the INTR bit.

Read/Write accessible always. STINTE is set to 0 by H_RESET and is not affected by S_RESET or setting the STOP bit

9 MREINT MII Management Read Error Interrupt. The MII Read Error interrupt is set by the Am79C972 controller to indicate that the currently read register from the external PHY is invalid. The contents of BCR34 are incorrect and that the operation should be performed again. The indication of an incorrect read comes from the PHY. During the read turnaround time of the MII management frame the external PHY should drive the MDIO pin to a LOW state. If this does not happen, it indicates that the PHY and the Am79C972 controller have lost synchronization.

When MREINT is set to 1, $\overline{\text{INTA}}$ is asserted if the enable bit MREINTE is set to 1.

Read/Write accessible always. MREINT is cleared by the host by writing a 1. Writing a 0 has no effect. MREINT is cleared by H_RESET and is not affected by S_RESET or setting the STOP bit.

8 MREINTE MII Management Read Error Interrupt Enable. If MREINTE is set, the MREINT bit will be able to set the INTR bit.

Read/Write accessible always. MREINTE is set to 0 by

H_RESET and is not affected by S_RESET or setting the STOP bit

7 MAPINT MII Management Auto-Poll Interrupt. The MII Auto-Poll interrupt is set by the Am79C972 controller to indicate that the currently read status does not match the stored previous status indicating a change in state for the external PHY. A change in the Auto-Poll Access Method (BCR32, Bit 11) will reset the shadow register and will not cause an interrupt on the first access from the Auto-Poll section. Subsequent accesses will generate an interrupt if the shadow register and the read register produce differences.

When MAPINT is set to 1, $\overline{\text{INTA}}$ is asserted if the enable bit MAPINTE is set to 1.

Read/Write accessible always. MAPINT is cleared by the host by writing a 1. Writing a 0 has no effect. MAPINT is cleared by H_RESET and is not affected by S_RESET or setting the STOP bit.

6 MAPINTE MII Auto-Poll Interrupt Enable. If MAPINTE is set, the MAPINT bit will be able to set the INTR bit.

Read/Write accessible always. MAPINTE is set to 0 by H_RESET and is not affected by S_RESET or setting the STOP bit

5 MCCINT MII Management Command Complete Interrupt. The MII Management Command Complete Interrupt is set by the Am79C972 controller when a read or write operation to the MII Data Port (BCR34) is complete.

When MCCINT is set to 1, $\overline{\text{INTA}}$ is asserted if the enable bit MCCINTE is set to 1.

Read/Write accessible always. MCCINT is cleared by the host by writing a 1. Writing a 0 has no effect. MCCINT is cleared by H_RESET and is not affected by S_RESET or setting the STOP bit.

4 MCCINTE MII Management Command Complete Interrupt Enable. If MCCINTE is set to 1, the MCCINT bit will be able to set the INTR bit when the host reads or writes to the MII Data Port (BCR34) only. Internal MII Management Commands will not generate an interrupt. For instance Auto-Poll state machine generated MII management frames will not generate an interrupt upon completion unless there is a compare error which get reported through the MAPINT (CSR7, bit 6) interrupt or the MCCIINTE is set to 1.

Read/Write accessible always. MCCINTE is set to 0 by H_RESET and is not affected by S_RESET or setting the STOP bit.

3 MCCIINT MII Management Command Complete Internal Interrupt. The MII Management Command Complete Interrupt is set by the Am79C972 controller when a read or write operation on the MII management port is complete from an internal operation. Examples of internal operations are Auto-Poll or MII Management Port generated MII management frames. These are normally hidden to the host.

When MCCIINT is set to 1, $\overline{\text{INTA}}$ is asserted if the enable bit MCCINTE is set to 1.

Read/Write accessible always. MCCIINT is cleared by the host by writing a 1. Writing a 0 has no effect. MCCIINT is cleared by H_RESET and is not affected by S_RESET or setting the STOP bit.

2 MCCIINTE MII Management Command Complete Internal Interrupt Enable. If MCCIINTE is set to 1, the MCCIINT bit will be able to set the INTR bit when the internal state machines generate MII

		management frames. For instance, when MCCIINTE is set to 1 and the Auto-Poll state machine generates a MII management frame, the MCCIINT will set the INTR bit upon completion of the MII management frame regardless of the comparison outcome. Read/Write accessible always. MCCIINTE is set to 0 by H_RESET and is not affected by S_RESET or setting the STOP bit.
1	MIIPDTINT	MII PHY Detect Transition Interrupt. The MII PHY Detect Transition Interrupt is set by the Am79C972 controller whenever the MIIPD bit (BCR32, bit 14) transitions from 0 to 1 or vice versa. Read/Write accessible always. MIIPDTINT is cleared by the host by writing a 1. Writing a 0 has no effect. MIIPDTINT is cleared by H_RESET and is not affected by S_RESET or setting the STOP bit.
0	MIIPDTINTE	MII PHY Detect Transition Interrupt Enable. If MIIPDTINTE is set to 1, the MIIPDTINT bit will be able to set the INTR bit. Read/Write accessible always. MIIPDTINTE is set to 0 by H_RESET and is not affected by S_RESET or setting the STOP bit.

CSR8: Logical Address Filter 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 register write has been performed on this register. Read/Write accessible only when either the STOP or the SPND bit is set. These bits are unaffected by H_RESET, S_RESET, or STOP.

CSR9: Logical Address Filter 1

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 register write has been performed on this register. Read/Write accessible only when either the STOP or the SPND bit is set. These bits are unaffected by H_RESET, S_RESET, or STOP.

CSR10: Logical Address Filter 2

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 register write has been performed on this register. Read/Write accessible only when either the STOP or the SPND bit is set. These bits are unaffected by H_RESET, S_RESET, or STOP.

CSR11: Logical Address Filter 3

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 register

write has been performed on this register.

Read/Write accessible only when either the STOP or the SPND bit is set. These bits are unaffected by H_RESET, S_RESET, or STOP.

CSR12: Physical Address Register 0

Note: *Bits 15-0 in this register are programmable through the EEPROM.*

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 contents of this register are loaded from EEPROM after H_RESET or by an EEPROM read command (PRGAD, BCR19, bit 14). If the EEPROM is not present, the contents of this register are undefined. This register can also be loaded from the initialization block after the INIT bit in CSR0 has been set or a direct register write has been performed on this register. Read/Write accessible only when either the STOP or the SPND bit is set. These bits are unaffected by H_RESET, S_RESET, or STOP.

CSR13: Physical Address Register 1

Note: *Bits 15-0 in this register are programmable through the EEPROM.*

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 contents of this register are loaded from EEPROM after H_RESET or by an EEPROM read command (PRGAD, BCR19, bit 14). If the EEPROM is not present, the contents of this register are undefined. This register can also be loaded from the initialization block after the INIT bit in CSR0 has been set or a direct register write has been performed on this register. Read/Write accessible only when either the STOP or the SPND bit is set. These bits are unaffected by H_RESET, S_RESET, or STOP.

CSR14: Physical Address Register 2

Note: *Bits 15-0 in this register are programmable through the EEPROM.*

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 contents of this register are loaded from EEPROM after H_RESET or by an EEPROM read command (PRGAD, BCR19, bit 14). If the EEPROM is not present, the contents of this register are undefined. This register can also be loaded from the initialization block after the INIT bit in CSR0 has been set or a direct register write has been performed on this register. Read/Write accessible only when either the STOP or the SPND bit is set. These bits are unaffected by H_RESET, S_RESET, or STOP.

CSR15: Mode

This register's fields are loaded during the Am79C972 controller initialization routine with the corresponding Initialization Block values, or when a direct register write has been performed on this register.

Bit	Name	Description
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 either the STOP or the SPND bit is set.

14 DRCVBC Disable Receive Broadcast. When set, disables the Am79C972 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 either the STOP or the SPND bit is set.

13 DRCVPA Disable Receive Physical Address. When set, the physical address detection (Station or node ID) of the Am79C972 controller will be disabled. Frames addressed to the nodes individual physical address will not be recognized.

Read/Write accessible only when either the STOP or the SPND bit is set.

12-9 RES Reserved locations. Written as zeros and read as undefined.

8-7 PORTSEL[1:0] Port Select bits allow for software controlled selection of the network medium. The only legal values for this field are 11 and 10.

A value of 11 selects the MII port and a value of 10 selects the GPSI port.

Read/Write accessible only when either the STOP or the SPND 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, bit 2).

Read/Write accessible only when either the STOP or the SPND bit is set.

5 DRTY Disable Retry. When DRTY is set to 1, the Am79C972 controller will attempt only one transmission. In this mode, the device will not protect the first 64 bytes of frame data in the Transmit FIFO from being overwritten, because automatic retransmission will not be necessary. When DRTY is set to 0, the Am79C972 controller will attempt 16 transmissions before signaling a retry error.

Read/Write accessible only when either the STOP or the SPND bit is set.

4 FCOLL Force Collision. This bit allows the collision logic to be tested. The Am79C972 controller must be in internal loopback for FCOLL to be valid. If FCOLL = 1, a collision will be forced during loopback transmission attempts, which will result in a Retry Error. If FCOLL = 0, the Force Collision logic will be disabled. FCOLL is defined after the initialization block is read.

Read/Write accessible only when either the STOP or the SPND bit is set.

3 DXMTFCS Disable Transmit CRC (FCS). When DXMTFCS is set to 0, the transmitter will generate and append an FCS to the transmitted frame. When DXMTFCS is set to 1, no FCS is generated or sent with the transmitted frame. DXMTFCS is overridden when ADD_FCS and ENP bits are set in TMD1.

When APAD_XMT bit (CSR4, bit11) is set to 1, the setting of DXMTFCS has no effect on frames shorter than 64 bytes.

If DXMTFCS is set and ADD_FCS is clear for a particular frame, no FCS will be generated. 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. See also the ADD_FCS bit in TMD1.

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

Read/Write accessible only when either the STOP or the SPND bit is set.

2 LOOP Loopback Enable allows the Am79C972 controller to operate in full-duplex mode for test purposes. The setting of the full-duplex control bits in BCR9 have no effect when the device operates in loopback mode. When LOOP = 1, loopback is enabled. In combination with INTL and MIIILP, various loopback modes are defined as follows in Table 21.

Table 21. Loopback Configuration

	LOOP	INTL	MIIILP	Function
GPSI	0	0	0	Normal Operation
	1	1	0	Internal Loop
	1	0	0	External Loop
MII	0	0	0	Normal Operation
	0	0	1	Internal Loop
	1	0	0	External Loop

Refer to *Loop Back Operation* section for more details.

Read/Write accessible only when either the STOP or the SPND bit is set. LOOP is cleared by H_RESET or S_RESET and is unaffected by STOP.

1 DTX Disable Transmit results in Am79C972 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 either the STOP or the SPND bit is set.

0 DRX Disable Receiver results in the Am79C972 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 either the STOP or the SPND 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	IADRL	This register is an alias of CSR1. Read/Write accessible only when either the STOP or the SPND bit is set.

CSR17: Initialization Block Address Upper

Bit	Name	Description
31-16	RES	Reserved locations. Written as zeros and read as undefined.
15-0	IADRH	This register is an alias of CSR2. Read/Write accessible only when either the STOP or the SPND bit is set.

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 Am79C972 controller will store incoming frame data. Read/Write accessible only when either the STOP or the SPND 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 Am79C972 controller will store incoming frame data.

		Read/Write accessible only when either the STOP or the SPND 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 Am79C972 controller is transmitting. Read/Write accessible only when either the STOP or the SPND 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 Am79C972 controller is transmitting. Read/Write accessible only when either the STOP or the SPND 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 Am79C972 controller will store incoming frame data. Read/Write accessible only when either the STOP or the SPND 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 Am79C972 controller will store incoming frame data. Read/Write accessible only when either the STOP or the SPND 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 either the STOP or the SPND 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 either the STOP or the SPND 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 receive descriptor address pointer.

Read/Write accessible only when either the STOP or the SPND 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 receive descriptor address pointer. Read/Write accessible only when either the STOP or the SPND 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 receive descriptor address pointer. Read/Write accessible only when either the STOP or the SPND 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 receive descriptor address pointer. Read/Write accessible only when either the STOP or the SPND 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 either the STOP or the SPND 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 either the STOP or the SPND 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 transmit descriptor address pointer. Read/Write accessible only when either the STOP or the SPND 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 transmit descriptor address pointer.

		Read/Write accessible only when either the STOP or the SPND 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 transmit descriptor address pointer. Read/Write accessible only when either the STOP or the SPND 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 transmit descriptor address pointer. Read/Write accessible only when either the STOP or the SPND 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 either the STOP or the SPND 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 either the STOP or the SPND 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 either the STOP or the SPND 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 either the STOP or the SPND 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 zeros.
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 either the STOP or the SPND 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.
15-0	CRST	Current Receive Status. This field is a copy of bits 31-16 of RMD1 of the current receive descriptor. Read/Write accessible only when either the STOP or the SPND bit is set. These bits are unaffected by H_RESET, S_RESET, or STOP.

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 zeros.
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 either the STOP or the SPND 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.
15-0	CXST	Current Transmit Status. This field is a copy of bits 31-16 of TMD1 of the current transmit descriptor. Read/Write accessible only when either the STOP or the SPND bit is set. These bits are unaffected by H_RESET, S_RESET, or STOP.

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 zeros.
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 either the STOP or the SPND 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.
15-0	NRST	Next Receive Status. This field is a copy of bits 31-16 of RMD1 of the next receive descriptor. Read/Write accessible only when either the STOP or the SPND bit is set. These bits are unaffected by H_RESET, S_RESET, or STOP.

CSR46: Transmit Poll Time Counter

Bit	Name	Description
31-16	RES	Reserved locations. Written as zeros and read as undefined.
15-0	TXPOLL	Transmit Poll Time Counter. This counter is incremented by the Am79C972 controller microcode and is used to trigger the transmit descriptor ring polling operation of the Am79C972 controller. Read/Write accessible only when either the STOP or the SPND bit is set. These bits are unaffected by H_RESET, S_RESET, or STOP.

CSR47: Transmit Polling Interval

Bit	Name	Description
31-16	RES	Reserved locations. Written as zeros and read as undefined.
15-0	TXPOLLINT	Transmit Polling Interval. This register contains the time that the Am79C972 controller will wait between successive polling operations. The TXPOLLINT value is expressed as the two's complement of the desired interval, where each bit of TXPOLLINT represents 1 clock period of time. TXPOLLINT[3:0] are ignored. (TXPOLLINT[16] is implied to be a one, so TXPOLLINT[15] is significant and does not represent the sign of the two's complement TXPOLLINT value.) The default value of this register is 0000h. This corresponds to a polling interval of 65,536 clock periods (1.966 ms when CLK = 33 MHz). The TXPOLLINT value of 0000h 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 (CSR0, bit 2). Then the user may write to CSR47 and then set STRT in CSR0. In this way, the default value of 0000h 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 writes all zeros to CSR47 as part of the alternative initialization sequence. Read/Write accessible only when either the STOP or the SPND bit is set. These bits are unaffected by H_RESET, S_RESET, or STOP.

CSR48: Receive Poll Time Counter

Bit	Name	Description
31-16	RES	Reserved locations. Written as zeros and read as undefined.
15-0	RXPOLL	Receive Poll Time Counter. This counter is incremented by the Am79C972 controller microcode and is used to trigger the receive descriptor ring polling operation of the Am79C972 controller. Read/Write accessible only when either the STOP or the SPND bit is set. These bits are unaffected by H_RESET, S_RESET, or STOP.

CSR49: Receive Polling Interval

Bit	Name	Description
31-16	RES	Reserved locations. Written as zeros and read as undefined.
15-0	RXPOLLINT	Receive Polling Interval. This register contains the time that the Am79C972 controller will wait between successive polling operations. The RXPOLLINT value is expressed as the two's comple-

ment of the desired interval, where each bit of RXPOLLINT approximately represents one clock time period. RXPOLLINT[3:0] are ignored. (RXPOLLINT[16] is implied to be a 1, so RXPOLLINT[15] is significant and does not represent the sign of the two's complement RXPOLLINT value.)

The default value of this register is 0000h. This corresponds to a polling interval of 65,536 clock periods (1.966 ms when CLK = 33 MHz). The RXPOLLINT value of 0000h is created during the microcode initialization routine and, therefore, might not be seen when reading CSR49 after H_RESET or S_RESET.

If the user desires to program a value for RXPOLLINT 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 (CSR0, bit 2). Then the user may write to CSR49 and then set STRT in CSR0. In this way, the default value of 0000h 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 writes all zeros to CSR49 as part of the alternative initialization sequence.

Read/Write accessible only when either the STOP or the SPND bit is set. These bits are unaffected by H_RESET, S_RESET, or STOP.

CSR58: Software Style

This register is an alias of the location BCR20. Accesses to and from this register are equivalent to accesses to BCR20.

Bit	Name	Description
31-16	RES	Reserved locations. Written as zeros and read as undefined.
15-11	RES	Reserved locations. Written as zeros and read as undefined.
10	APERREN	Advanced Parity Error Handling Enable. When APERREN is set to 1, the BPE bits (RMD1 and TMD1, bit 23) start having a meaning. BPE will be set in the descriptor associated with the buffer that was accessed when a data parity error occurred. Note that since the advanced parity error handling uses an additional bit in the descriptor, SWSTYLE (bits 7-0 of this register) must be set to 2 or 3 to program the Am79C972 controller to use 32-bit software structures. APERREN does not affect the reporting of address parity errors or data parity errors that occur when the Am79C972 controller is the target of the transfer. Read anytime, write accessible only when either the STOP or the SPND bit is set. APERREN is cleared by H_RESET and is not affected by S_RESET or STOP.
9	RES	Reserved locations. Written as zeros and read as undefined.
8	SSIZE32	Software Size 32 bits. When set, this bit indicates that the Am79C972 controller utilizes 32-bit software structures for the initialization block and the transmit and receive descriptor entries. When cleared, this bit indicates that the Am79C972 controller utilizes 16-bit software structures for the initialization block and the transmit and receive descriptor entries. In this mode, the Am79C972 controller is backwards compatible with the

Am7990 LANCE and Am79C960 PCnet-ISA controllers.

The value of SSIZE32 is determined by the Am79C972 controller according to the setting of the Software Style (SWSTYLE, bits 7-0 of this register).

Read accessible always. SSIZE32 is read only; write operations will be ignored. SSIZE32 will be cleared after H_RESET (since SWSTYLE defaults to 0) 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 Am79C972 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 the Am79C972 controller bus master accesses.

If SSIZE32 is set, then the software structures that are common to the Am79C972 controller and the host system will supply a full 32 bits for each address pointer that is needed by the Am79C972 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 (BCR18, bit 7).

7-0	SWSTYLE	Software Style register. The value in this register determines the style of register and memory resources that shall be used by the Am79C972 controller. The Software Style selection will affect the interpretation of a few bits within the CSR space, the order of the descriptor entries and the width of the descriptors and initialization block entries. All Am79C972 controller CSR bits and BCR bits and all descriptor, buffer, and initialization block entries not cited in Table 22 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 either the STOP or the SPND bit is set. The SWSTYLE register will contain the value 00h following H_RESET and will be unaffected by S_RESET or STOP.

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 transmit descriptor address pointer. The Am79C972 controller has the capability to stack multiple transmit frames. Read/Write accessible only when either the STOP or the SPND bit is set. These bits are unaffected by H_RESET, S_RESET, or STOP.

Table 22. Software Styles

SWSTYLE [7:0]	Style Name	SSIZE32	Initialization Block Entries	Descriptor Ring Entries
00h	LANCE/ PCnet-ISA controller	0	16-bit software structures, non-burst or burst access	16-bit software structures, non-burst access only
01h	RES	1	RES	RES
02h	PCnet-PCI controller	1	32-bit software structures, non-burst or burst access	32-bit software structures, non-burst access only
03h	PCnet-PCI controller	1	32-bit software structures, non-burst or burst access	32-bit software structures, non-burst or burst access
All Other	Reserved	Undefined	Undefined	Undefined

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 transmit descriptor address pointer. The Am79C972 controller has the capability to stack multiple transmit frames. Read/Write accessible only when either the STOP or the SPND 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.
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 either the STOP or the SPND bit is set. These bits are unaffected by H_RESET, S_RESET, or STOP.

CSR63: Previous Transmit Status

Bit	Name	Description
31-16	RES	Reserved locations. Written as zeros and read as undefined.
15-0	PXST	Previous Transmit Status. This field is a copy of bits 31-16 of TMD1 of the previous transmit descriptor. Read/Write accessible only when either the STOP or the SPND bit is set. These bits are unaffected by H_RESET, S_RESET, or STOP.

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 Am79C972 controller will transmit an outgoing frame. Read/Write accessible only when either the STOP or the SPND bit is set. These bits are unaffected by H_RESET, S_RESET, or STOP.