PRELIMINARY

# Am79C974

Advanced
Micro
Devices

## PCnet$^{TM}$-SCSI Combination Ethernet and SCSI Controller for PCI Systems

## DISTINCTIVE CHARACTERISTICS

### PCI Features

- **Direct glueless interface to 33 MHz, 32-bit PCI local bus**
- **132 Mbyte/s burst DMA transfer rate**
- **Compliant to PCI local bus Specification Revision 2.0**

### Ethernet Features

- **Supports ISO 8802-3 (IEEE/ANSI 802.3) and Ethernet Standards**
- **High-performance Bus Master architecture with integrated DMA Buffer Management Unit for low CPU and bus utilization**
- **Individual 136-byte transmit and 128-byte receive FIFOs provide frame buffering for increased system latency**
- **Microwire$^{TM}$ EEPROM interface supports jumperless design**
- **Integrated Manchester Encoder/Decoder**
- **Provides integrated Attachment Unit Interface (AUI) and 10BASE-T transceiver with automatic port selection**
- **Automatic Twisted-Pair receive polarity detection and automatic correction of the receive polarity**
- **Dynamic transmit FCS generation programmable on a frame-by-frame basis**
- **Internal/external loopback capabilities**
- **Supports the following types of network interfaces:**
  - — AUI to external 10BASE2, 10BASE5, 10BASE-T or 10BASE-F MAU
  - — Internal 10BASE-T transceiver with Smart Squelch to Twisted-Pair medium

### SCSI Features

- **Compliant to ANSI standards X3.131 – 1986 (SCSI-1) and X3.131 – 199X (SCSI-2)**
- **Fast 8-bit SCSI-2 10 Mbyte/s synchronous or 7 Mbyte/s asynchronous data transfer rate**
- **SCSI specific Bus Mastering DMA engine (32-bit address/data)**
- **96-byte DMA FIFO for low bus latency**
- **On-chip state machine to control the SCSI sequences in hardware**
- **Integrated industry standard Fast SCSI-2 core**
- **Single-Ended 48 mA outputs to drive the SCSI bus directly**
- **Support for Scatter-Gather DMA data transfers**
- **Hooks in silicon and software to enable disk drive spin down for power savings**

### General Features

- **Software compatible with AMD's Am79C960 PCnet-ISA, Am79C961 PCnet-ISA+, Am79C965 PCnet-32, Am79C970 PCnet-PCI register and descriptor architecture**
- **Plug-in and software compatible with AMD's PCSCSI family of SCSI controllers for PCI**
- **NAND Tree test mode for connectivity testing on printed circuit boards**
- **Single +5 V power supply operation**
- **Low-power, CMOS design with sleep modes for both Ethernet and SCSI controllers allows reduced power consumption for critical battery powered applications and 'Green PCs'**
- **Fully static design for low frequency and power operation**
- **132-pin PQFP package**

## GENERAL DESCRIPTION

The PCnet-SCSI combination Ethernet and 8-bit Fast SCSI controller with a 32-bit PCI bus interface is a highly integrated Ethernet-Fast SCSI system solution designed to address high-performance system application requirements. This single-chip is a flexible bus-mastering device that can be used in many applications, including network- and SCSI-ready PCs, printers, fax modems, and bridge/router designs. The bus-master architecture provides high data throughput in the system and low CPU and system bus utilization. The PCnet-SCSI controller is fabricated with AMD's advanced low-power CMOS process to provide low operating and standby current for power sensitive applications.

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

Publication# **18681** Rev. **B** Amendment **/1**
Issue Date: **October 1994**

The PCnet-SCSI is part of AMD's PCI product family of plug-in and software compatible SCSI and Ethernet controllers. This product compatibility ensures a low cost system upgrade path and lower motherboard manufacturing costs.

## Ethernet Specific

The PCnet-SCSI controller includes a complete Ethernet node integrated into a single VLSI device. It contains a bus interface unit, a DMA buffer management unit, an IEEE 802.3-defined Media Access Control (MAC) function, individual 136-byte transmit and 128-byte receive FIFOs, an IEEE 802.3-defined Attachment Unit Interface (AUI) and Twisted-Pair Transceiver Media Attachment Unit (10BASE-T MAU), and a Microwire EEPROM interface. The PCnet-SCSI controller is also register compatible with the LANCE (Am7990) Ethernet controller, the C-LANCE (Am79C90) Ethernet controller, the ILACC (Am79C900) Ethernet controller, and all Ethernet controllers in the PCnet Family, including the PCnet-ISA controller (Am79C960), the PCnet-ISA+ controller (Am79C961), and the PCnet-32 controller (Am79C965). The buffer management unit supports the LANCE, ILACC, and PCnet descriptor software models. The PCnet-SCSI controller is software compatible with the Novell NE2100 and NE1500 Ethernet adapter card architectures. In addition, a Sleep function has been incorporated to provide low standby current, excellent for notebooks and Green PCs.

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

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

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

## SCSI Specific

The PCnet-SCSI controller also includes a high-performance Fast SCSI controller with a glueless interface to the PCI local bus. The PCnet-SCSI integrates its own 32-bit bus mastering DMA engine with an industry standard Fast SCSI-2 block. The DMA engine and accompanying 96 byte DMA FIFO allow 32-bit burst data transfers across the high bandwidth PCI bus at speeds of up to 132 Mbyte/s. Full support for scatter-gather DMA transfers optimize performance in multi-tasking system applications.

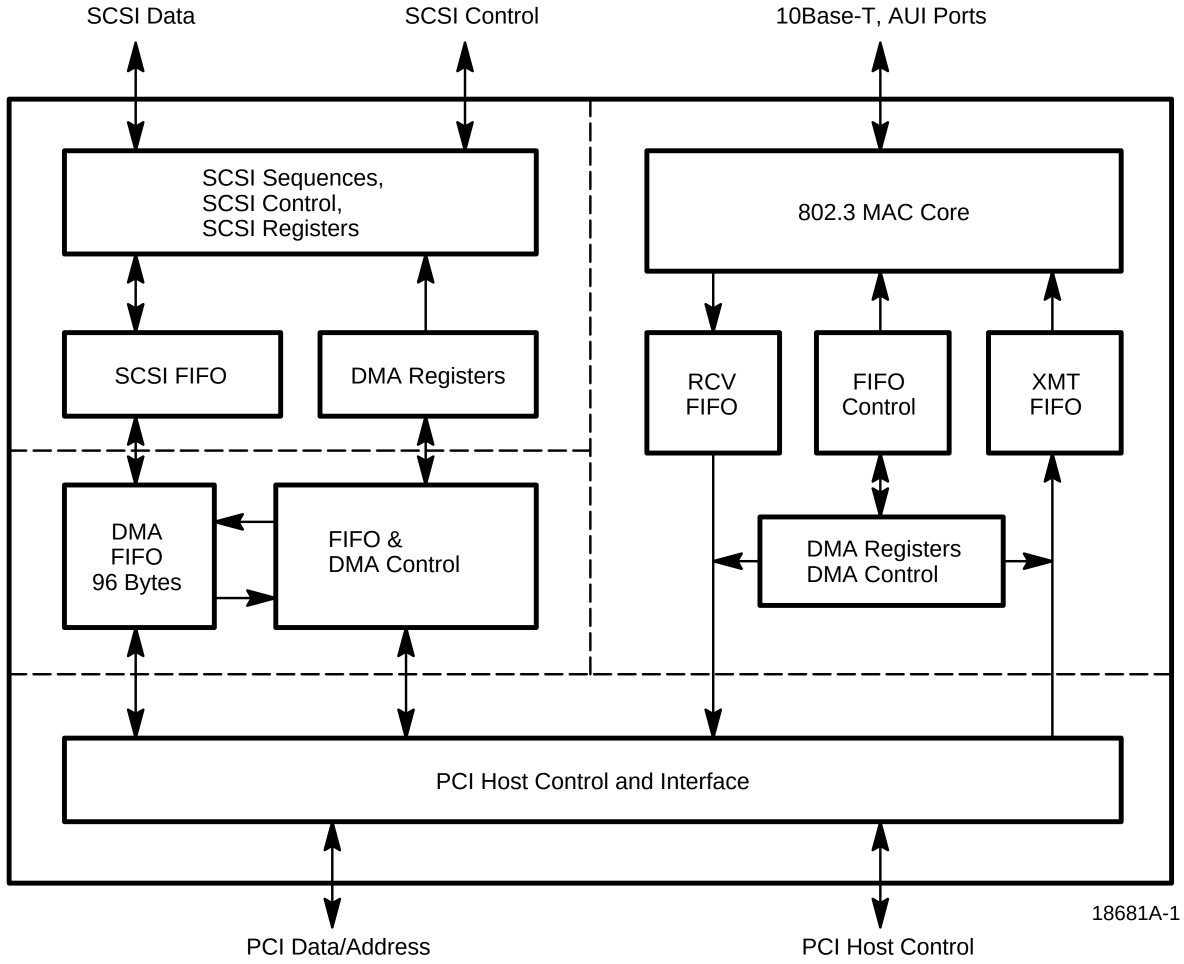
The PCnet-SCSI's on-chip state machine controls SCSI bus sequences in hardware and is coupled with the bus mastering DMA engine to eliminate the need for an on-chip RISC processor. This results in a smaller die size giving the Am79C974 superior price/performance versus competitive offerings.

AMD supports the Am79C974 with a total system solution which includes:

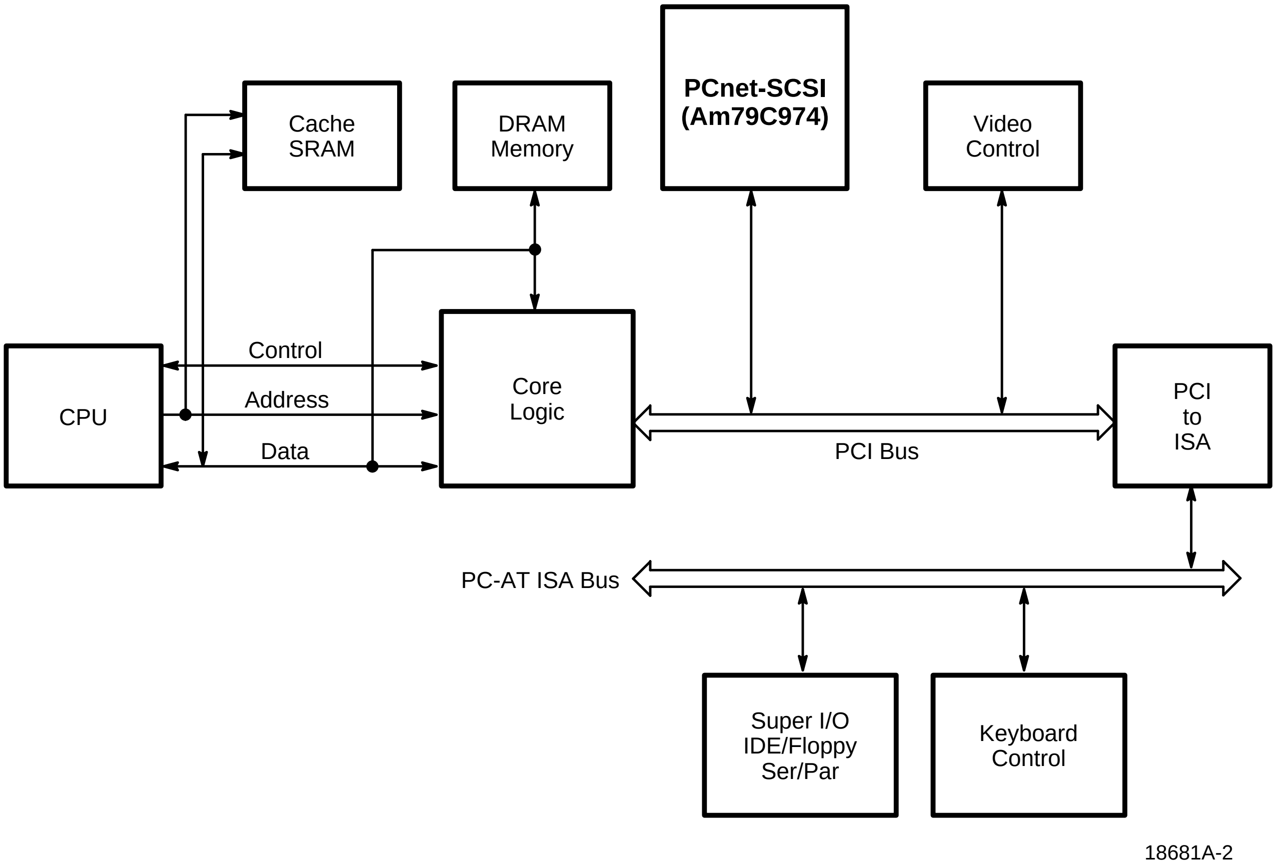
- A full suite of licensable SCSI drivers and utilities fully tested under the following operating system environments:
  - — DOS 5.0 – 6.0
  - — Windows 3.1
  - — Windows NT
  - — OS/2 2.x
  - — Netware 3.x, 4.x
  - — SCO UNIX 3.2.4, ODT 2.0
- An INT13h Compatible SCSI ROM BIOS
- ASPI Compatibility
- Complete hardware reference design kit

For more detailed information on the PCnet-SCSI refer to the technical manual, PID #18738A.

## HIGH LEVEL BLOCK DIAGRAM



18681A-1



**Am79C974 in a PCI System**

## TABLE OF CONTENTS

Ethernet Controller

SCSI Controller

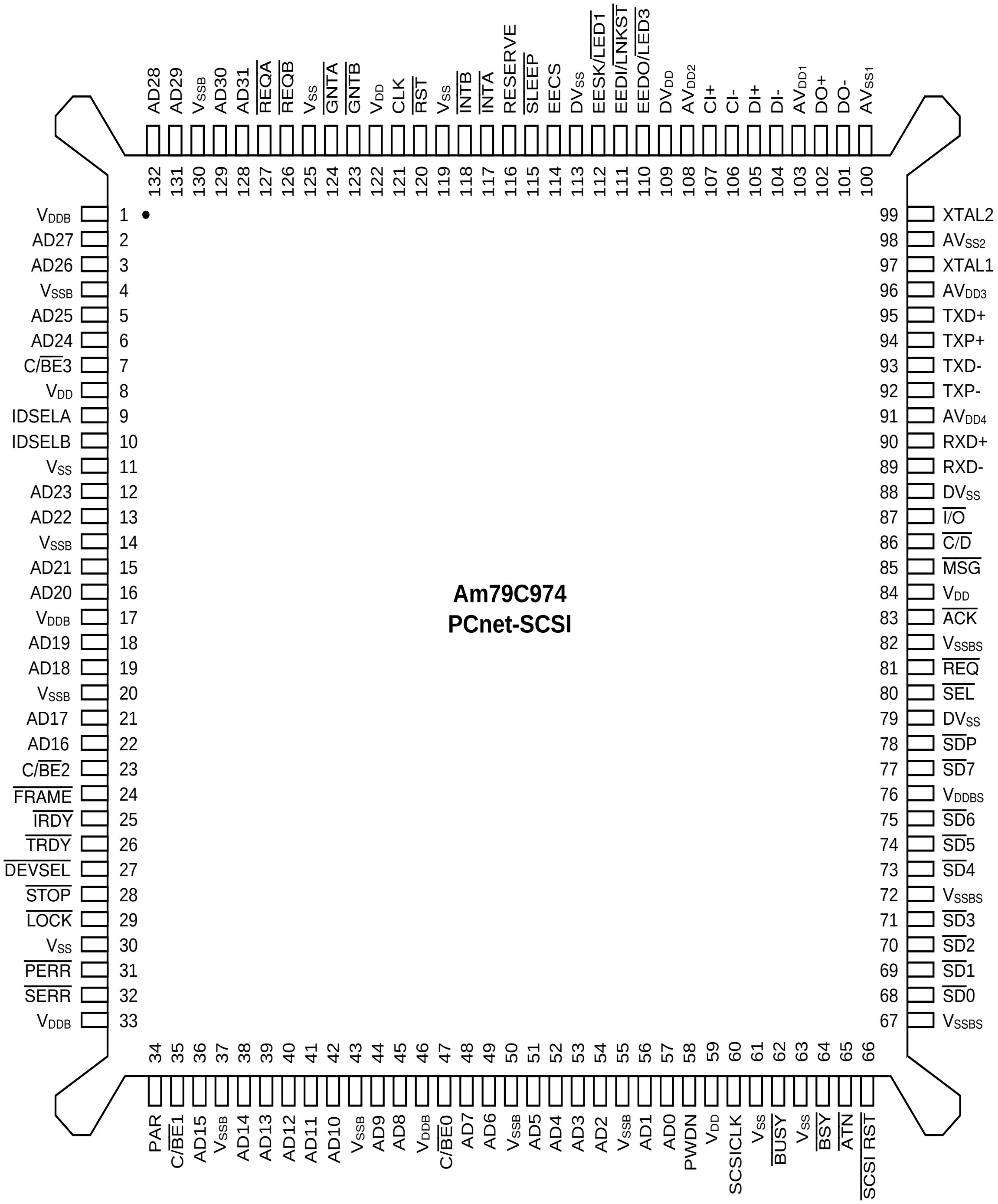
## LIST OF FIGURES

## LIST OF TABLES

## RELATED PRODUCTS

| Part No. | Description |
|---|---|
| Am33C93A | Synchronous SCSI Controller |
| Am386® | High-Performance 32-Bit Microprocessor |
| Am486™ | High-Performance 32-Bit Microprocessor |
| Am53C94/96 | High-Performance SCSI Controller |
| Am53C974 | PCSCSI™ Bus Mastering Fast SCSI Controller for PCI Systems |
| Am53CF94/96 | Enhanced Fast SCSI-2 Controller |
| Am79C90 | CMOS Local Area Network Controller for Ethernet (C-LANCE) |
| Am79C98 | Twisted-Pair Ethernet Transceiver (TPEX) |
| Am79C100 | Twisted-Pair Ethernet Transceiver Plus (TPEX+) |
| Am79C900 | Integrated Local Area Communications Controller™ (ILACC™) |
| Am79C940 | Media Acces Controller for Ethernet (MACE™) |
| Am79C960 | PCnet-ISA Single-Chip Ethernet Controller (for ISA bus) |
| Am79C961 | PCnet-ISA+ Single-Chip Ethernet Controller (with Microsoft® Plug n' Play support) |
| Am79C965 | PCnet-32 Single-Chip 32-Bit Ethernet Controller (for 386DX, 486 and VL buses) |
| Am79C970 | PCnet-PCI Single-Chip Ethernet Controller for PCI Local Bus |
| Am79C981 | Integrated Multiport Repeater Plus™ (IMR+™) |
| Am79C987 | Hardware Implemented Management Information Base™ (HIMIB™) |
| Am7990 | Local Area Network Controller for Ethernet (LANCE) |
| Am7996 | IEEE 802.3/Ethernet/Cheapernet Tap Transceiver |
| Am85C30 | Enhanced Serial Communication Controller |

## CONNECTION DIAGRAM



18681A-3

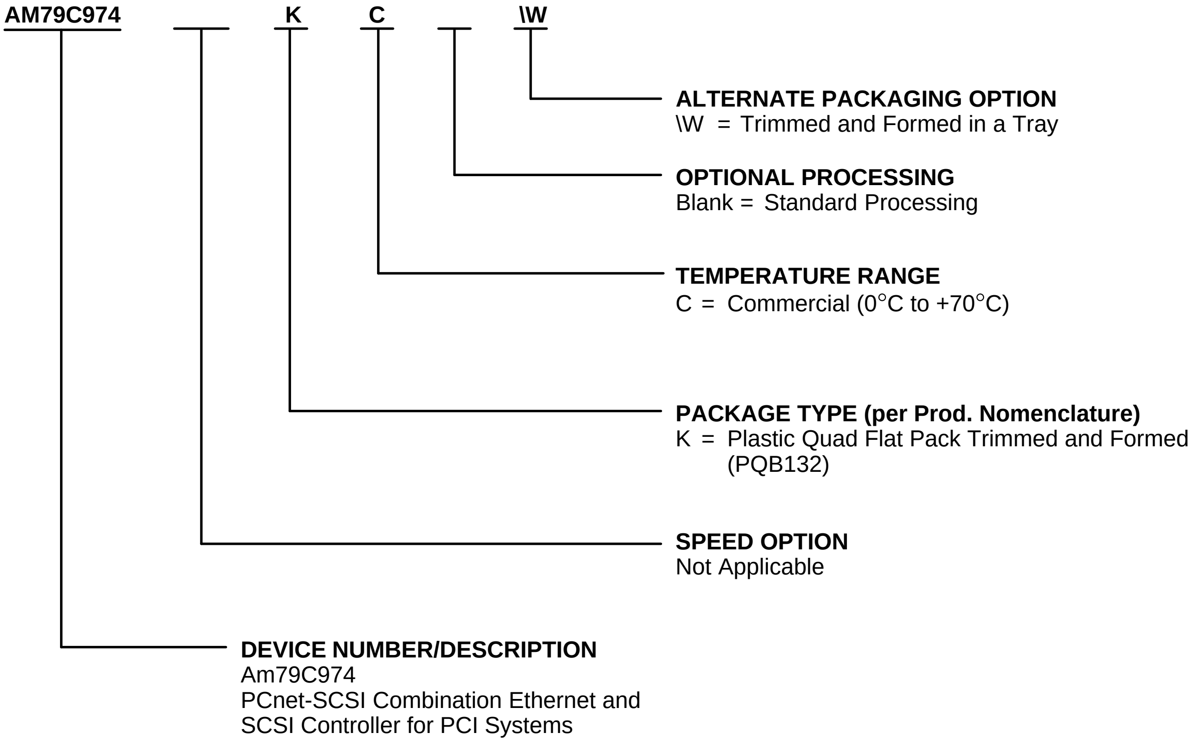
*Pin 1 is marked for orientation.*

*RESERVE = Don't Connect.*

# ORDERING INFORMATION

## Standard Products

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



AM79C974 ____ K C ____ \W

**ALTERNATE PACKAGING OPTION**
\W = Trimmed and Formed in a Tray

**OPTIONAL PROCESSING**
Blank = Standard Processing

**TEMPERATURE RANGE**
C = Commercial (0°C to +70°C)

**PACKAGE TYPE (per Prod. Nomenclature)**
K = Plastic Quad Flat Pack Trimmed and Formed (PQB132)

**SPEED OPTION**
Not Applicable

**DEVICE NUMBER/DESCRIPTION**
Am79C974
PCnet-SCSI Combination Ethernet and
SCSI Controller for PCI Systems

| Valid Combinations | |
|---|---|
| AM79C974 | KC\W |

**Valid Combinations**

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

## PIN DESIGNATIONS

### Listed by Pin Number

| Pin No. | Pin Name | Pin No. | Pin Name | Pin No. | Pin Name | Pin No. | Pin Name |
|---|---|---|---|---|---|---|---|
| 1 | $V_{DDB}$ | 34 | PAR | 67 | $V_{SSBS}$ | 100 | $AV_{SS1}$ |
| 2 | AD27 | 35 | C/$\overline{\text{BE}}$1 | 68 | $\overline{\text{SD}}$0 | 101 | DO– |
| 3 | AD26 | 36 | AD15 | 69 | $\overline{\text{SD}}$1 | 102 | DO+ |
| 4 | $V_{SSB}$ | 37 | $V_{SSB}$ | 70 | $\overline{\text{SD}}$2 | 103 | $AV_{DD1}$ |
| 5 | AD25 | 38 | AD14 | 71 | $\overline{\text{SD}}$3 | 104 | DI– |
| 6 | AD24 | 39 | AD13 | 72 | $V_{SSBS}$ | 105 | DI+ |
| 7 | C/$\overline{\text{BE}}$3 | 40 | AD12 | 73 | $\overline{\text{SD}}$4 | 106 | CI– |
| 8 | $V_{DD}$ | 41 | AD11 | 74 | $\overline{\text{SD}}$5 | 107 | CI+ |
| 9 | IDSELA | 42 | AD10 | 75 | $\overline{\text{SD}}$6 | 108 | $AV_{DD2}$ |
| 10 | IDSELB | 43 | $V_{SSB}$ | 76 | $V_{DDBS}$ | 109 | $DV_{DD}$ |
| 11 | $V_{SS}$ | 44 | AD9 | 77 | $\overline{\text{SD}}$7 | 110 | EEDO/$\overline{\text{LED3}}$ |
| 12 | AD23 | 45 | AD8 | 78 | $\overline{\text{SD}}$P | 111 | EEDI/$\overline{\text{LNKST}}$ |
| 13 | AD22 | 46 | $V_{DDB}$ | 79 | $DV_{SS}$ | 112 | EESK/$\overline{\text{LED1}}$ |
| 14 | $V_{SSB}$ | 47 | C/$\overline{\text{BE}}$0 | 80 | $\overline{\text{SEL}}$ | 113 | $DV_{SS}$ |
| 15 | AD21 | 48 | AD7 | 81 | $\overline{\text{REQ}}$ | 114 | EECS |
| 16 | AD20 | 49 | AD6 | 82 | $V_{SSBS}$ | 115 | $\overline{\text{SLEEP}}$ |
| 17 | $V_{DDB}$ | 50 | $V_{SSB}$ | 83 | $\overline{\text{ACK}}$ | 116 | RESERVE |
| 18 | AD19 | 51 | AD5 | 84 | $DV_{DD}$ | 117 | $\overline{\text{INTA}}$ |
| 19 | AD18 | 52 | AD4 | 85 | $\overline{\text{MSG}}$ | 118 | $\overline{\text{INTB}}$ |
| 20 | $V_{SSB}$ | 53 | AD3 | 86 | $\overline{\text{C/D}}$ | 119 | $V_{SS}$ |
| 21 | AD17 | 54 | AD2 | 87 | $\overline{\text{I/O}}$ | 120 | $\overline{\text{RST}}$ |
| 22 | AD16 | 55 | $V_{SSB}$ | 88 | $DV_{SS}$ | 121 | CLK |
| 23 | C/$\overline{\text{BE}}$2 | 56 | AD1 | 89 | RXD– | 122 | $V_{DD}$ |
| 24 | $\overline{\text{FRAME}}$ | 57 | AD0 | 90 | RXD+ | 123 | $\overline{\text{GNTB}}$ |
| 25 | $\overline{\text{IRDY}}$ | 58 | PWDN | 91 | $AV_{DD4}$ | 124 | $\overline{\text{GNTA}}$ |
| 26 | $\overline{\text{TRDY}}$ | 59 | $V_{DD}$ | 92 | TXP– | 125 | $V_{SS}$ |
| 27 | $\overline{\text{DEVSEL}}$ | 60 | SCSICLK | 93 | TXD– | 126 | $\overline{\text{REQB}}$ |
| 28 | $\overline{\text{STOP}}$ | 61 | $V_{SS}$ | 94 | TXP+ | 127 | $\overline{\text{REQA}}$ |
| 29 | $\overline{\text{LOCK}}$ | 62 | $\overline{\text{BUSY}}$ | 95 | TXD+ | 128 | AD31 |
| 30 | $V_{SS}$ | 63 | $V_{SS}$ | 96 | $AV_{DD3}$ | 129 | AD30 |
| 31 | $\overline{\text{PERR}}$ | 64 | $\overline{\text{BSY}}$ | 97 | XTAL1 | 130 | $V_{SSB}$ |
| 32 | $\overline{\text{SERR}}$ | 65 | $\overline{\text{ATN}}$ | 98 | $AV_{SS2}$ | 131 | AD29 |
| 33 | $V_{DDB}$ | 66 | $\overline{\text{SCSI RST}}$ | 99 | XTAL2 | 132 | AD28 |

## PIN DESIGNATIONS

### Listed by Pin Name

| Pin Name | Pin No. | Pin Name | Pin No. | Pin Name | Pin No. | Pin Name | Pin No. |
|---|---|---|---|---|---|---|---|
| $\overline{\text{ACK}}$ | 83 | $\overline{\text{ATN}}$ | 65 | $\overline{\text{GNTB}}$ | 123 | $\overline{\text{STOP}}$ | 28 |
| AD0 | 57 | $AV_{DD1}$ | 103 | IDSELA | 9 | $\overline{\text{TRDY}}$ | 26 |
| AD1 | 56 | $AV_{DD2}$ | 108 | IDSEL | 10 | XTAL1 | 97 |
| AD2 | 54 | $AV_{DD3}$ | 96 | $\overline{\text{INTA}}$ | 117 | XTAL2 | 99 |
| AD3 | 53 | $AV_{DD4}$ | 91 | $\overline{\text{INTB}}$ | 118 | TXD– | 93 |
| AD4 | 52 | $AV_{SS1}$ | 100 | $\overline{\text{I/O}}$ | 87 | TXD+ | 95 |
| AD5 | 51 | $AV_{SS2}$ | 98 | $\overline{\text{IRDY}}$ | 25 | TXP– | 92 |
| AD6 | 49 | $\overline{\text{BSY}}$ | 64 | $\overline{\text{LOCK}}$ | 29 | TXP+ | 94 |
| AD7 | 48 | $\overline{\text{BUSY}}$ | 62 | $\overline{\text{MSG}}$ | 85 | $V_{DD}$ | 8 |
| AD8 | 45 | C/$\overline{\text{BE}}$0 | 47 | PAR | 34 | $V_{DD}$ | 59 |
| AD9 | 44 | C/$\overline{\text{BE}}$1 | 35 | $\overline{\text{PERR}}$ | 31 | $V_{DD}$ | 122 |
| AD10 | 42 | C/$\overline{\text{BE}}$2 | 23 | PWDN | 58 | $V_{DDB}$ | 1 |
| AD11 | 41 | C/$\overline{\text{BE}}$3 | 7 | $\overline{\text{REQ}}$ | 81 | $V_{DDB}$ | 17 |
| AD12 | 40 | $\overline{\text{C/D}}$ | 86 | $\overline{\text{REQA}}$ | 127 | $V_{DDB}$ | 33 |
| AD13 | 39 | CLK | 121 | $\overline{\text{REQB}}$ | 126 | $V_{DDB}$ | 46 |
| AD14 | 38 | CI– | 106 | RESERVE | 116 | $V_{DDBS}$ | 76 |
| AD15 | 36 | CI+ | 107 | $\overline{\text{RST}}$ | 120 | $V_{SS}$ | 11 |
| AD16 | 22 | $\overline{\text{DEVSEL}}$ | 27 | RXD– | 89 | $V_{SS}$ | 30 |
| AD17 | 21 | DI– | 104 | RXD+ | 90 | $V_{SS}$ | 61 |
| AD18 | 19 | DI+ | 105 | SCSICLK | 60 | $V_{SS}$ | 63 |
| AD19 | 18 | DO– | 101 | $\overline{\text{SCSI RST}}$ | 66 | $V_{SS}$ | 119 |
| AD20 | 16 | DO+ | 102 | $\overline{\text{SD}}$0 | 68 | $V_{SS}$ | 125 |
| AD21 | 15 | $DV_{DD}$ | 84 | $\overline{\text{SD}}$1 | 69 | $V_{SSB}$ | 4 |
| AD22 | 13 | $DV_{DD}$ | 109 | $\overline{\text{SD}}$2 | 70 | $V_{SSB}$ | 14 |
| AD23 | 12 | $DV_{SS}$ | 79 | $\overline{\text{SD}}$3 | 71 | $V_{SSB}$ | 20 |
| AD24 | 6 | $DV_{SS}$ | 88 | $\overline{\text{SD}}$4 | 73 | $V_{SSB}$ | 37 |
| AD25 | 5 | $DV_{SS}$ | 113 | $\overline{\text{SD}}$5 | 74 | $V_{SSB}$ | 43 |
| AD26 | 3 | EECS | 114 | $\overline{\text{SD}}$6 | 75 | $V_{SSB}$ | 50 |
| AD27 | 2 | EEDI/$\overline{\text{LNKST}}$ | 111 | $\overline{\text{SD}}$7 | 77 | $V_{SSB}$ | 55 |
| AD28 | 132 | EEDO/$\overline{\text{LED3}}$ | 110 | $\overline{\text{SD}}$P | 78 | $V_{SSB}$ | 130 |
| AD29 | 131 | EESK/$\overline{\text{LED1}}$ | 112 | $\overline{\text{SEL}}$ | 80 | $V_{SSBS}$ | 67 |
| AD30 | 129 | $\overline{\text{FRAME}}$ | 24 | $\overline{\text{SERR}}$ | 32 | $V_{SSBS}$ | 72 |
| AD31 | 128 | $\overline{\text{GNTA}}$ | 124 | $\overline{\text{SLEEP}}$ | 115 | $V_{SSBS}$ | 82 |

# PIN DESIGNATIONS

## Quick Reference Pin Description

| Pin Name | Description | Type | Driver | # Pins |
|---|---|---|---|---|
| **PCI Bus Interface** | | | | |
| AD[31:00] | Address/Data Bus | IO | TS3 | 32 |
| C/$\overline{\text{BE}}$[3:0] | Bus Command/Byte Enable | IO | TS3 | 4 |
| CLK | Bus Clock | I | NA | 1 |
| $\overline{\text{DEVSEL}}$ | Device Select | IO | TS6 | 1 |
| $\overline{\text{FRAME}}$ | Cycle Frame | IO | TS6 | 1 |
| $\overline{\text{GNTA}}$, $\overline{\text{GNTB}}$ | Bus Grant | I | NA | 1 |
| IDSELA, IDSELB | Initialization Device Select | I | NA | 1 |
| $\overline{\text{INTA}}$, $\overline{\text{INTB}}$ | Interrupt | IO | OD6 | 1 |
| $\overline{\text{IRDY}}$ | Initiator Ready | IO | TS6 | 1 |
| $\overline{\text{LOCK}}$ | Bus Lock | IO | TS6 | 1 |
| PAR | Parity | IO | TS6 | 1 |
| $\overline{\text{PERR}}$ | Parity Error | IO | TS6 | 1 |
| $\overline{\text{REQA}}$, $\overline{\text{REQB}}$ | Bus Request | IO | TS3 | 1 |
| $\overline{\text{RST}}$ | Reset | I | NA | 1 |
| $\overline{\text{SERR}}$ | System Error | IO | OD6 | 1 |
| $\overline{\text{STOP}}$ | Stop | IO | TS6 | 1 |
| $\overline{\text{TRDY}}$ | Target Ready | IO | TS6 | 1 |
| **ETHERNET SPECIFIC**<br>**Board Interface** | | | | |
| EECS | Microwire Serial PROM Chip Select | O | O8 | 1 |
| EEDI/$\overline{\text{LNKST}}$ | Microwire Serial EEPROM Data In/Link Status | O | LED | 1 |
| EEDO/$\overline{\text{LED3}}$ | Microwire APROM Data Out/LED predriver | IO | LED | 1 |
| EESK/$\overline{\text{LED1}}$ | Microwire Serial PROM Clock/LED1 | IO | LED | 1 |
| SLEEP | Sleep Mode | I | NA | 1 |
| XTAL1–2 | Crystal Input/Output | IO | NA | 2 |
| **Attachment Unit Interface (AUI)** | | | | |
| CI+/CI– | AUI Collision Differential Pair | I | NA | 2 |
| DI+/DI– | AUI Data In Differential Pair | I | NA | 2 |
| DO+/DO– | AUI Data Out Differential Pair | O | DO | 2 |
| **10BASE-T Interface** | | | | |
| RXD+/RXD– | Receive Differential Pair | I | NA | 2 |
| TXD+/TXD– | Transmit Differential Pair | O | TDO | 2 |
| TXP+/TXP– | Transmit Pre-distortion Differential Pair | O | TPO | 2 |
| $\overline{\text{LNKST}}$/EEDI | Link Status/Microwire Serial EEPROM Data In | O | LED | 1 |

## PIN DESIGNATIONS (continued)

### Quick Reference Pin Description

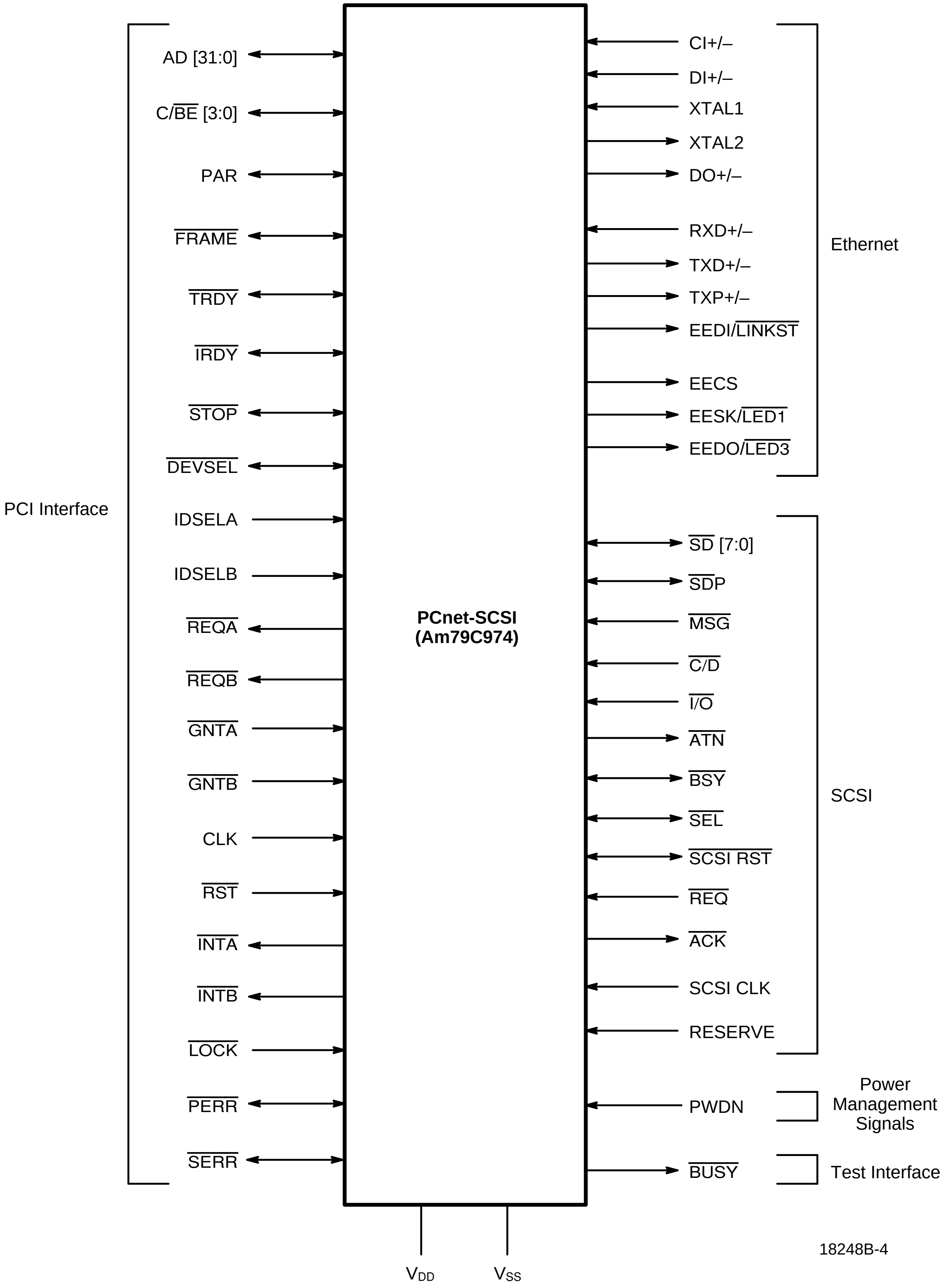
| Pin Name | Description | Type | Driver | # Pins |
|---|---|---|---|---|
| **SCSI SPECIFIC** | | | | |
| **SCSI Interface** | | | | |
| $\overline{\text{SD}}$ [7:0] | SCSI Data | IO | OD48 | 8 |
| $\overline{\text{SDP}}$ | SCSI Data Parity | IO | OD48 | 1 |
| $\overline{\text{MSG}}$ | Message | I | | 1 |
| $\overline{\text{C/D}}$ | Command/Data | I | | 1 |
| $\overline{\text{I/O}}$ | Input/Output | I | | 1 |
| $\overline{\text{ATN}}$ | Attention | O | OD48 | 1 |
| $\overline{\text{BSY}}$ | Busy | IO | OD48 | 1 |
| $\overline{\text{SEL}}$ | Select | IO | OD48 | 1 |
| $\overline{\text{SCSI RST}}$ | SCSI Bus Reset | IO | OD48 | 1 |
| $\overline{\text{REQ}}$ | Request | I | | 1 |
| $\overline{\text{ACK}}$ | Acknowledge | O | OD48 | 1 |
| **Miscellaneous** | | | | |
| SCSI CLK | SCSI Core Clock | I | | 1 |
| RESERVE | Reserved, DO NOT CONNECT | I | | 1 |
| **Power Management** | | | | |
| PWDN | Power Down Indicator | I | | 1 |
| **Test Interface** | | | | |
| $\overline{\text{BUSY}}$ | NAND Tree Test Output | O | O3 | 1 |
| **Power Supplies** | | | | |
| AV$_{DD}$ | Analog Power | P | NA | 4 |
| AV$_{SS}$ | Analog Ground | P | NA | 2 |
| V$_{DD}$/DV$_{DD}$ | Digital Power | P | NA | 5 |
| V$_{SS}$/DV$_{SS}$ | Digital Ground | P | NA | 9 |
| V$_{DDB}$/V$_{DDBS}$ | I/O Buffer Power | P | NA | 5 |
| V$_{SSB}$/V$_{SSBS}$ | I/O Buffer Ground | P | NA | 11 |

### Listed by Driver Type

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

| Name | Type | I$_{OL}$ (mA) | I$_{OH}$ (mA) | pF |
|---|---|---|---|---|
| TS3 | Tri-State$^{TM}$ | 3 | –2.0 | 50 |
| TS6 | Tri-State | 6 | –2.0 | 50 |
| O3 | Totem Pole | 3 | –0.4 | 50 |
| O6 | Totem Pole | 6 | –0.4 | 50 |
| O8 | Totem Pole | 8 | –0.4 | 50 |
| OD6 | Open Drain | 6 | NA | 50 |
| OD48 | Open Drain | 48 | NA | — |
| LED | LED | 12 | –0.4 | 50 |

## LOGIC SYMBOL



# PIN DESCRIPTION

## PCI Bus Interface

### AD[31:00]

**Address and Data**
**Input/Output, Active High**

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

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

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

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

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

**Bus Command and Byte Enables**
**Input/Output, Active Low**

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

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

### CLK

**Clock**
**Input**

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

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

### $\overline{\text{DEVSEL}}$

**Device Select**
**Input/Output, Active Low**

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

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

### $\overline{\text{FRAME}}$

**Cycle Frame**
**Input/Output, Active Low**

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

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

### $\overline{\text{GNTA}}$

**Bus Grant**
**Input, Active Low**

This signal indicates that the access to the bus has been granted to the Am79C974’s SCSI controller.

The Am79C974 controller supports bus parking. When the PCI bus is idle and the system arbiter asserts $\overline{\text{GNTA}}$ without an active $\overline{\text{REQA}}$ from the Am79C974 controller, the controller will actively drive the AD[31:00], C/$\overline{\text{BE}}$[3:0], and PAR lines.

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

### $\overline{\text{GNTB}}$

**Bus Grant**
**Input, Active Low**

This signal indicates that the access to the bus has been granted to the Am79C974’s Ethernet controller. The Am79C974 controller supports bus parking. When the PCI bus is idle and the system arbiter asserts $\overline{\text{GNTB}}$ without an active $\overline{\text{REQB}}$ from the Am79C974 controller, the controller will actively drive the AD, C/$\overline{\text{BE}}$ and PAR lines.

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

## IDSELA

**Initialization Device Select**
**Input, Active High**

This signal is used as a SCSI controller selection for the Am79C974 during configuration read and write transaction.

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

## IDSELB

**Initialization Device Select**
**Input, Active High**

This signal is used as an Ethernet controller selection for the PCnet-SCSI controller during configuration read and write transaction.

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

## $\overline{\text{INTA}}$

**Interrupt Request**
**Input/Output, Active Low, Open Drain**

This signal combines the interrupt requests from both the SCSI DMA engine and the SCSI core. The interrupt source can be determined by reading the SCSI DMA Status Register. It is cleared when the Status Register is read.

When $\overline{\text{RST}}$ is active, $\overline{\text{INTA}}$ is an input for NAND tree testing. This is the only time $\overline{\text{INTA}}$ is an input.

## $\overline{\text{INTB}}$

**Interrupt Request**
**Input/Output, Active Low, Open Drain**

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

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

When $\overline{\text{RST}}$ is active, $\overline{\text{INTB}}$ is an input for NAND tree testing. This is the only time $\overline{\text{INTB}}$ is an input.

## $\overline{\text{IRDY}}$

**Initiator Ready**
**Input/Output, Active Low**

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

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

## $\overline{\text{LOCK}}$

**Lock**
**Input, Active Low**

$\overline{\text{LOCK}}$ is used by the current bus master to indicate an atomic operation that may require multiple transfers.

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

The PCnet-SCSI controller will never assert $\overline{\text{LOCK}}$ as a master.

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

## PAR

**Parity**
**Input/Output, Active High**

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

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

## $\overline{\text{PERR}}$

**Parity Error**
**Input/Output, Active Low, Open Drain**

This signal is asserted for one CLK by the PCnet-SCSI controller when it detects a parity error during any data phase when its AD[31:00] lines are inputs. The $\overline{\text{PERR}}$ pin is only active when PERREN (bit 6) in the PCI command register is set.

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

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

### $\overline{\text{REQA}}$

**Bus Request**
**Input/Output, Active Low**

The Am79C974's SCSI controller asserts $\overline{\text{REQA}}$ pin as a signal that it wishes to become a bus master. Once asserted, $\overline{\text{REQA}}$ remains active until $\overline{\text{GNTA}}$ has become active.

When $\overline{\text{RST}}$ is active, $\overline{\text{REQA}}$ is an input for NAND tree testing. This is the only time $\overline{\text{REQA}}$ is an input.

### $\overline{\text{REQB}}$

**Bus Request**
**Input/Output, Active Low**

The Am79C974's Ethernet controller asserts $\overline{\text{REQB}}$ pin as a signal that it wishes to become a bus master. Once asserted, $\overline{\text{REQB}}$ remains active until $\overline{\text{GNT}}$ has become active, independent of subsequent assertion of $\overline{\text{SLEEP}}$ or setting of the STOP bit or access to the S_RESET port (offset 14h).

When $\overline{\text{RST}}$ is active, $\overline{\text{REQB}}$ is an input for NAND tree testing. This is the only time $\overline{\text{REQB}}$ is an input.

### $\overline{\text{RST}}$

**Reset**
**Input, Active Low**

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

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

### $\overline{\text{SERR}}$

**System Error**
**Input/Output, Active Low, Open Drain**

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

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

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

### $\overline{\text{STOP}}$

**Stop**
**Input/Output, Active Low**

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

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

### $\overline{\text{TRDY}}$

**Target Ready**
**Input/Output, Active Low**

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

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

## Ethernet Controller Pins

**Board Interface**

### $\overline{\text{LED1}}$

**LED1**
**Output**

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

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

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.

### LED3

**LED3**
**Output**

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

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

### LNKST

**LINK Status**
**Output**

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

### SLEEP

**Sleep**
**Input**

When SLEEP is asserted (active LOW), the PCnet-SCSI controller performs an internal system reset of the S_RESET type and then proceeds into a power savings mode. (The reset operation caused by SLEEP assertion will not affect BCR registers.) The PCI interface section is not effected by SLEEP. In particular, access to the PCI configuration space remains possible. None of the configuration registers will be reset by SLEEP. All I/O accesses to the PCnet-SCSI controller will result in a PCI target abort response. The PCnet-SCSI controller will not assert REQ while in sleep mode. When SLEEP is asserted, all non-PCI interface outputs will be placed in their normal S_RESET condition. All non-PCI interface inputs will be ignored except for the SLEEP pin itself. De-assertion of SLEEP results in wake-up. The system must refrain from starting the network operations of the PCnet-SCSI device for 0.5 seconds following the deassertion of the SLEEP signal in order to allow internal analog circuits to stabilize.

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

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

The SLEEP pin does not affect the SCSI section.

### XTAL1

**Crystal Oscillator Input**
**Input**

### XTAL2

**Crystal Oscillator Output**
**Output**

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

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

#### Microwire EEPROM Interface

### EESK

**EEPROM Serial Clock**
**Input/Output**

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

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

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

## EEDO

**EEPROM Data Out**
**Input**

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

EEDO is shared with the LED3 function.

## EECS

**EEPROM Chip Select**
**Output**

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

## EEDI

**EEPROM Data In**
**Output**

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

EEDI is shared with the LNKST function.

### Attachment Unit Interface

## CI±

**Collision In**
**Input**

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

## DI±

**Data In**
**Input**

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

## DO±

**Data Out**
**Output**

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

### Twisted-Pair Interface

## RXD±

**10BASE-T Receive Data**
**Input**

10BASE-T port differential receivers.

## TXD±

**10BASE-T Transmit Data**
**Output**

10BASE-T port differential drivers.

## TXP±

**10BASE-T Pre-Distortion Control**
**Output**

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

## SCSI Controller Pins

**SCSI Bus Interface Signals**
**SCSI Bus Pins**

### $\overline{\text{SD}}$ [7:0]

**SCSI Data**
**Input/Output, Active Low, Open Drain/Active Negation, Schmitt Trigger**
These pins are defined as bi-directional SCSI data bus.

### $\overline{\text{SDP}}$

**SCSI Data Parity**
**Input/Output, Active Low, Open Drain/Active Negation, Schmitt Trigger**
This pin is defined as bi-directional SCSI data parity.

### $\overline{\text{MSG}}$

**Message**
**Input, Active Low, Schmitt Trigger**
It is a Schmitt trigger input in the initiator mode.

### $\overline{\text{C/D}}$

**Command/Data**
**Input, Schmitt Trigger**
It is a Schmitt trigger input in the initiator mode.

### $\overline{\text{I/O}}$

**Input/Output**
**Input, Schmitt Trigger**
It is a Schmitt trigger input in the initiator mode.

### $\overline{\text{ATN}}$

**Attention**
**Output, Active Low, Open Drain**

This signal is a 48 mA output in the initiator mode. This signal will be asserted when the device detects a parity error; also, it can be asserted via certain commands.

### $\overline{\text{BSY}}$

**Busy**
**Input/Output, Active Low, Schmitt Trigger, Open Drain**
As a SCSI input signal it has a Schmitt trigger and as an output signal it has a 48 mA drive.

### $\overline{\text{SEL}}$

**Select**
**Input/Output, Active Low, Schmitt Trigger, Open Drain**
As a SCSI input signal it has a Schmitt trigger and as an output signal it has a 48 mA drive.

### $\overline{\text{SCSI RST}}$

**Reset**
**Input/Output, Active Low, Schmitt Trigger, Open Drain**
As a SCSI input signal it has a Schmitt trigger and as an output signal it has a 48 mA drive.

### $\overline{\text{REQ}}$

**Request**
**Input, Active Low, Schmitt Trigger**
This is a SCSI input signal with a Schmitt trigger in the initiator mode.

### $\overline{\text{ACK}}$

**Acknowledge**
**Output, Active Low, Open Drain/Active Negation**
This is a SCSI output signal with a 48 mA drive in the initiator mode.

### SCSI CLK

**SCSI Clock**
**Input**
The SCSI clock signal is used to generate all internal device timings. The maximum frequency of this input is 40 MHz and a minimum of 10 MHz is required to maintain the SCSI bus timings.

***Note:***
*A 40 MHz clock must be supplied at this input to achieve 10 Mbyte/s Synchronous Fast SCSI transfers.*

### PWDN

**Power Down Indicator**
**Input, Active High**
This signal, when asserted, sets the PWDN status bit in the DMA status register and sends an interrupt to the host.

## Test Interface

### $\overline{\text{BUSY}}$

**NAND Tree Out**
**Output, Active Low**
This signal is logically equivalent to the SCSI bus signal $\overline{\text{BSY}}$. It is duplicated so that external logic can be connected to monitor SCSI bus activity.

The results of the NAND tree testing can be observed on the $\overline{\text{BUSY}}$ pin where $\overline{\text{RST}}$ is asserted; otherwise, $\overline{\text{BUSY}}$ will reflect the state of the SCSI Bus Signal line $\overline{\text{BSY}}$ (pin 64).

## Miscellaneous

### RESERVED

**Reserved_DO NOT CONNECT**
**Input**

This pin (#116) is reserved for internal test logic. It **MUST NOT BE CONNECTED** to anything for proper chip operation. It's use is subject to change in future products.

## Power Supply Pins

## Analog Power Supply Pins

### AV$_{DD}$

**Analog Power (4 Pins)**
**Power**

There are four analog +5 V supply pins. Special attention should be paid to the printed circuit board layout to avoid excessive noise on these lines.  Refer to Appendix C and the *Technical Manual* (PID #18738A) for details.

### AV$_{SS}$

**Analog Ground (2 Pins)**
**Power**

There are two analog ground pins. Special attention should be paid to the printed circuit board layout to avoid excessive noise on these lines. Refer to Appendix C and the *Technical Manual* (PID #18738A) for details.

## Digital Power Supply Pins

### V$_{DD}$/DV$_{DD}$

**Digital Power (5 Pins)**
**Power**

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

### V$_{DDB}$/V$_{DDB}$

**I/O Buffer Power (5 Pins)**
**Power**

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

### V$_{SS}$/DV$_{SS}$

**Digital Ground (9 Pins)**
**Ground**

There are 9 ground pins that are used by the internal digital circuitry.

### V$_{SSB}$/V$_{SSBS}$

**I/O Buffer Ground (11 Pins)**
**Ground**

There are 11 ground pins that are used by the PCI bus Input/Output buffer drivers.

# BASIC FUNCTIONS

## System Bus Interface Function

During normal operations the Am79C974 operates as a bus master with a few slave I/O accesses for status and control functions.

The Ethernet controller is initialized through a combination of PCI Configuration Space accesses, I/O space Bus Slave accesses, Memory Space Bus Master accesses, and optional reads of an external serial EEPROM. The EEPROM is read through the Microwire interface either automatically by the Am79C974 or indirectly by a series of bus slave accesses to one of the Ethernet Bus Configuration Registers (BCRs). The EEPROM normally contains the ISO 8802-3 (IEEE/ANSI 802.3) Ethernet node address and data to be loaded into some of the Ethernet BCRs.

The SCSI controller is initialized by bus slave writes to SCSI Core and SCSI DMA registers.

## Software Interface

The Am79C794 uses four address spaces: Ethernet PCI configuration space, SCSI PCI configuration space, I/O space, and memory space.

SCSI PCI configuration space is selected when the IDSELA pin is active. Ethernet PCI configuration space is selected when the IDSELB pin is active. The way that IDSELA and IDSELB are controlled depends on external hardware. Section 3.6.4.1 of the PCI Specification recommends two methods of generating configuration cycles called Configuration Mechanism #1 and Configuration Mechanism #2.

The PCI Configuration Spaces are used by system software to identify the SCSI and Ethernet controllers and to set up device configuration without the use of jumpers. Certain PCI configuration registers have read-only information about the devices resource requirements. Other registers are used as mail boxes that system configuration software uses to inform other software what resources have been allocated to the device. The only PCI Configuration Registers that affect the operation of the Am79C794 are the SCSI and Ethernet Base Address Registers, which are found at offset 10h in each of the two configuration spaces, and the Command Registers at offset 4. Writing to these registers establishes the base address of the SCSI I/O space and the base address of the Ethernet I/O space.

The SCSI controller registers occupy 96 bytes of I/O space that starts on whatever 128-byte boundary that is programmed into the Base Address Register at offset 10h in the SCSI PCI Configuration Space. The Ethernet controller registers occupy 32 bytes of I/O space that starts on whatever 32-byte boundary that is programmed into the Base Address Register at offset 10h in the Ethernet PCI Configuration Space. These registers are used to set up controller operating modes, to enable or disable various features, to start certain operations, and to monitor operating status.

In addition to the registers in the I/O space, the Ethernet controller uses certain data structures that are set up (typically by the host computer) in normal memory space. These data structures are (1) the initialization block that contains configuration data that the Ethernet controller automatically loads into its Configuration and Status Registers (CSRs), (2) the Receive and Transmit Descriptor Rings, that contain pointers to receive and transmit buffers and status and control information about these buffers, and (3) the receive and transmit buffers. The Ethernet controller uses bus master accesses to read the locations of the buffers, to store frames received from the network into the receive buffers, and to transmit the contents of the transmit buffers.

## Ethernet Interfaces

The Am79C974 controller can be connected to an 802.3 network via one of two network interfaces. The Attachment Unit Interface (AUI) provides an ISO 8802-3 (IEEE/ANSI 802.3) compliant differential interface to a remote MAU or an on-board transceiver. The 10BASE-T interface provides a twisted-pair Ethernet port. While in auto-selection mode, the interface in use is determined by an auto-sensing mechanism which checks the link status on the 10BASE-T port. If there is no active link status, then the device assumes an AUI connection.

## SCSI Interfaces

The Am79C974 acts as a bridge between the PCI and SCSI buses. As the maximum data transfer rate on the PCI bus is a very high 132 Mbyte/s compared with the SCSI bus 10 Mbyte/s, buffering is required between the two buses. The buffering is provided by two FIFOs: a 16-byte (16X8 bits) SCSI Core FIFO and an additional 96-byte (24X32 bits) DMA FIFO. These FIFOs provide a temporary storage for all command, data, status and message bytes as they are transferred between the 32-bit PCI bus and the 8-bit SCSI bus.

The Am79C974's SCSI Core and DMA registers are addressed using the value in the Base Address Register (offset 10h in the PCI Configuration Space). The SCSI registers occupy 16 double words and the DMA engine registers occupy 8 double word locations. The I/O address map is as follows:

| Start Offset | End Offset | Block Name | Size |
|---|---|---|---|
| 0x0000 | 0x003F | SCSI Core Reg | 16 DW/64B |
| 0x0040 | 0x005F | PCI  DMA CCB | 8 DW/32B |

The PCI configuration space, Ethernet controller and SCSI controller are described in detail in the following sections.

## DETAILED FUNCTIONS

### Bus Interface Unit (BIU)

The bus interface unit is built of several state machines that run synchronously to CLK. One bus interface unit state machine handles accesses where the Am79C974 controller is the bus slave, and another handles accesses where the Am79C974 controller is the bus master. All inputs are synchronously sampled. All outputs are synchronously generated on the rising edge of CLK.

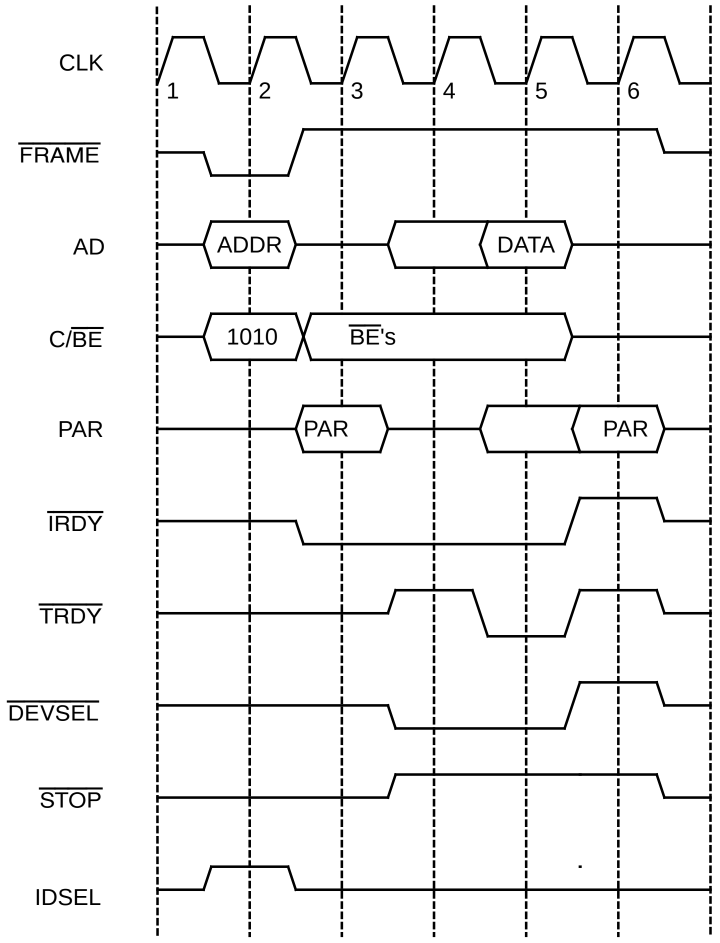
In the descriptions that follow, $\overline{\text{GNT}}$, $\overline{\text{REQ}}$, $\overline{\text{INT}}$, and IDSEL are used to refer to the set of $\overline{\text{GNTA}}$, $\overline{\text{REQA}}$, $\overline{\text{INTA}}$, and IDSELA for the SCSI controller and to the set of $\overline{\text{GNTB}}$, $\overline{\text{REQB}}$, $\overline{\text{INTB}}$, and IDSELB for the Ethernet Controller, respectively.

#### Slave Configuration Transfers

The host can access the Am79C974 PCI configuration space with a configuration read or write command. The Am79C974 controller will assert $\overline{\text{DEVSEL}}$ if the IDSEL input is asserted during the address phase and if the access is a configuration cycle. $\overline{\text{DEVSEL}}$ is asserted two clock cycles after the host has asserted $\overline{\text{FRAME}}$. All configuration cycles are of fixed length. The Am79C974 controller will assert $\overline{\text{TRDY}}$ on the 3rd clock of the data phase.

#### Slave Configuration Read

The Slave Configuration Read command is used by the host CPU to read the configuration space in the Am79C974 controller. This provides the host CPU with information concerning the device and its capabilities. This is a single cycle, non-burst 8-bit, 16-bit, or 32-bit transfer.

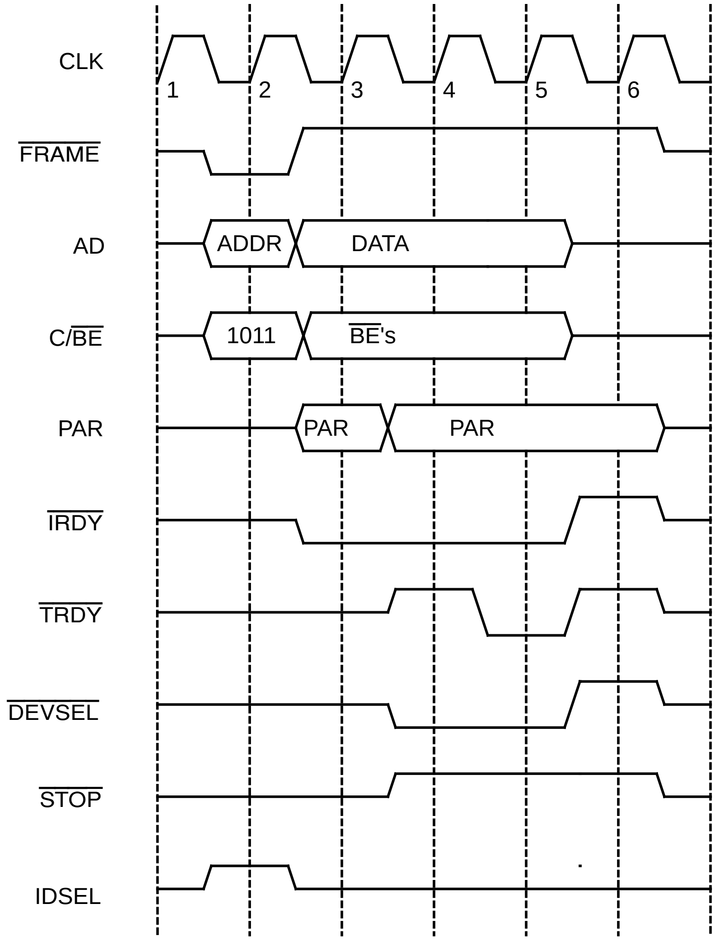


18681A-5

**Figure 1. Slave Configuration Read**

### Slave Configuration Write

The Slave Configuration Write command is used by the host CPU to write the configuration space in the Am79C974 controller. This allows the host CPU to control basic activity of the device, such as enable/disable, change I/O location, etc. This is a single cycle, non-burst 8-bit, 16-bit, or 32-bit transfer.



18681A-6

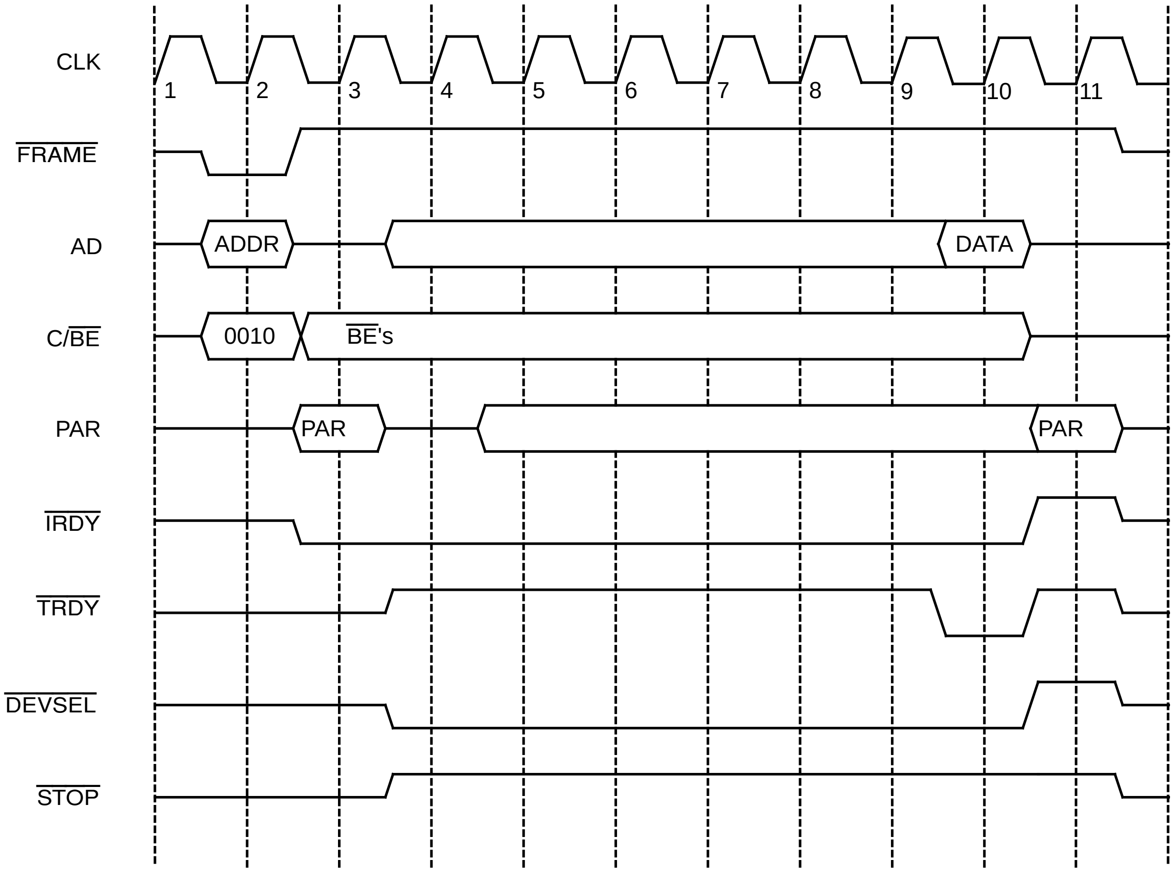
**Figure 2. Slave Configuration Write**

### Slave I/O Transfers

After the Am79C974 controller is configured as I/O device (by setting IOEN in the PCI Command register), it starts monitoring the PCI bus for access to its internal registers. The Am79C974 controller will look for an address that falls within its I/O address space. The Am79C974 controller will assert $\overline{\text{DEVSEL}}$ if it detects an address match and the access is an I/O cycle. $\overline{\text{DEVSEL}}$ is asserted two clock cycles after the host has asserted $\overline{\text{FRAME}}$. The Am79C974 controller will not assert $\overline{\text{DEVSEL}}$ if it detects an address match, but the PCI command is not of the type I/O read or I/O write. The Am79C974 controller will suspend looking for I/O cycles while being a bus master.

### Slave I/O Read

The Slave I/O Read command is used by the host CPU to read the Am79C974's CSRs, BCRs and EEPROM locations and SCSI and CCB registers. It is a single cycle, non-burst 8-bit,16-bit or 32-bit transfer which is initiated by the host CPU. The typical number of wait states added to a slave I/O read access on the part of the Am79C974 controller is 6 to 7 clock cycles. The Am79C974 controller will not produce Slave I/O Read commands while being a bus master.

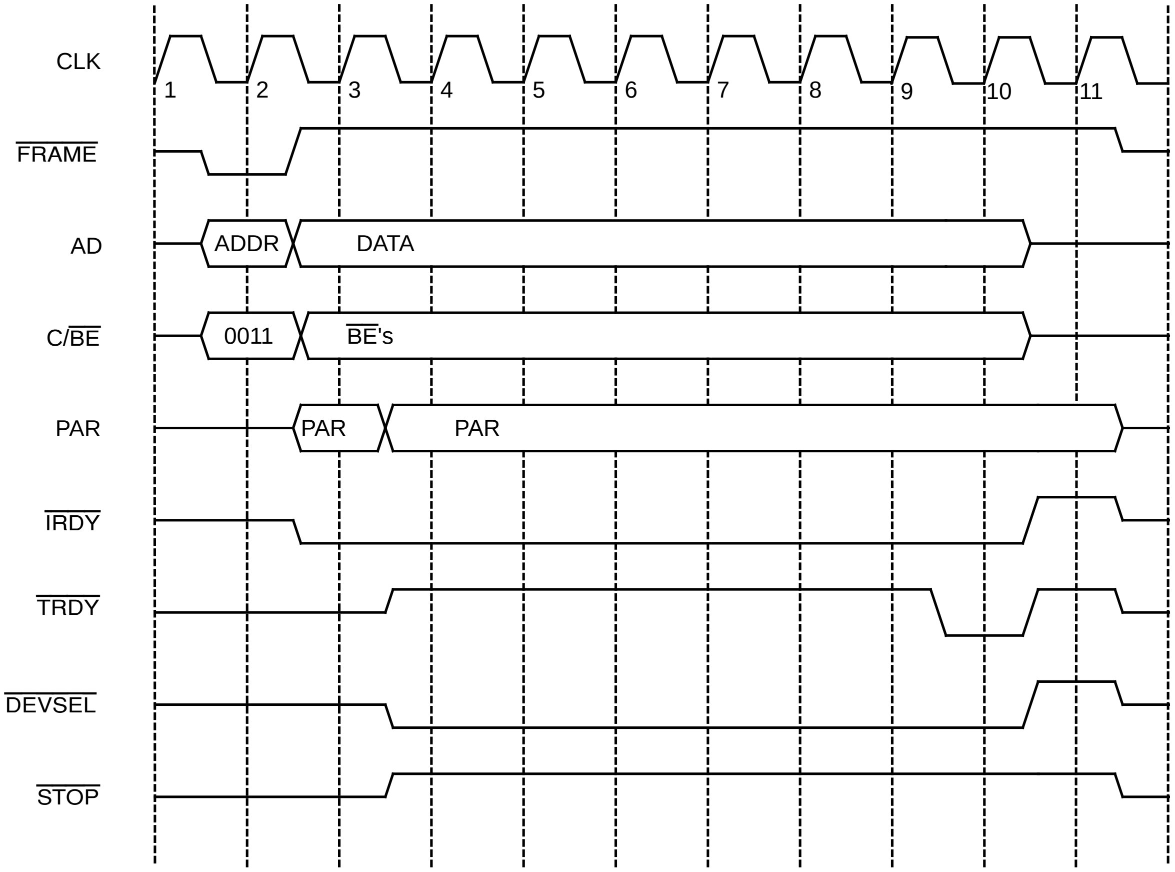


18681A-7

**Figure 3. Slave I/O Read**

### Slave I/O Write

The Slave I/O Write command is used by the host CPU to write to the Am79C974's CSRs, BCRs and EEPROM locations and SCSI and CCB registers. It is a single cycle, non-burst 8-bit, 16–bit, or 32-bit transfer which is initiated by the host CPU. The typical number of wait states added to a slave I/O write access on the part of the Am79C974 controller is 6 to 7 clock cycles. The Am79C974 controller will not produce Slave I/O write commands while being a bus master.



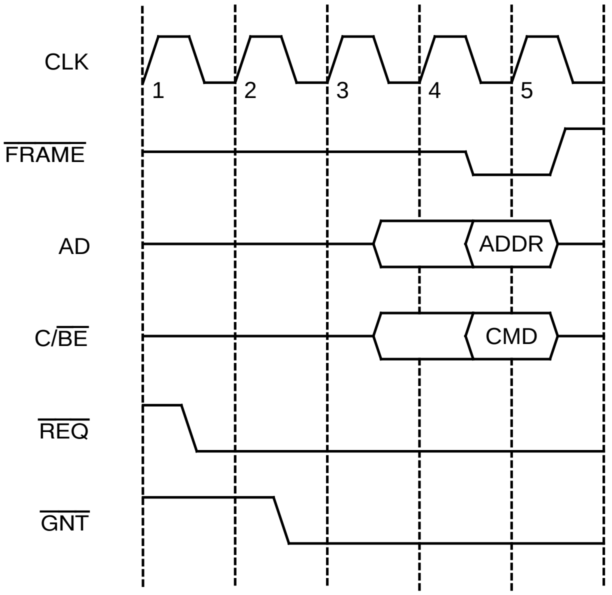
18681A-8

**Figure 4. Slave I/O Write**

### Bus Acquisition

The Am79C974 microcode (in the buffer management section) will determine when a DMA transfer should be initiated. The first step in any Am79C974 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 $\overline{\text{REQ}}$ signal and ownership is granted by the arbiter through the $\overline{\text{GNT}}$ signal.

Figure 5 shows the Am79C974 controller bus acquisition. $\overline{\text{GNT}}$ is asserted at clock 3. The Am79C974 controller starts driving AD[31:00] and C/$\overline{\text{BE}}$[3:0] prior to clock 4. $\overline{\text{FRAME}}$ is asserted at clock 5 indicating a valid address and command on AD[31:00] and C/$\overline{\text{BE}}$[3:0]. ADSTEP (bit 7) in the PCI Command register is set to ONE to indicated that the Am79C974 controller uses address stepping. Address stepping is only used for the first address phase of a bus master period.



18681A-9

**Figure 5. Bus Acquisition**

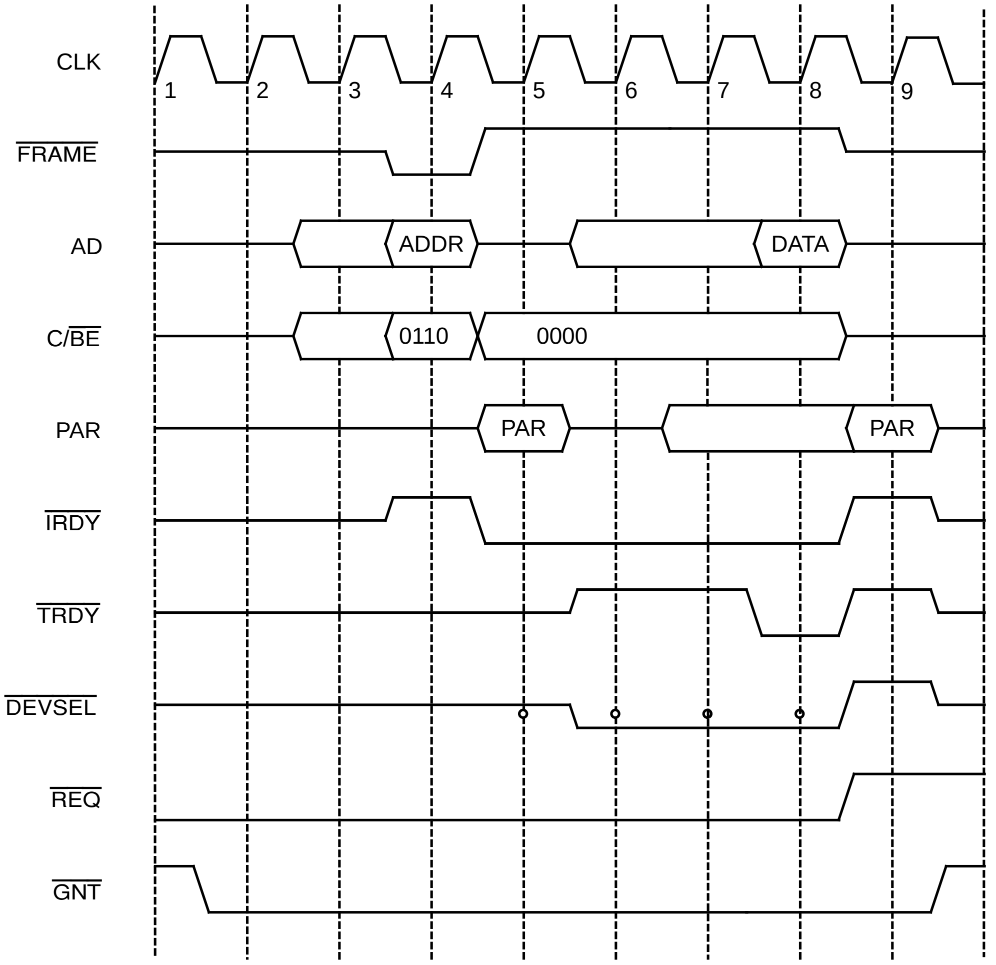
### Bus Master DMA Transfers

There are four primary types of DMA transfers. The Am79C974 controller uses non-burst as well as burst cycles for read and write access to the main memory.

### Basic Non-Burst Read Cycles

All Am79C974 controller non-burst read accesses are of the PCI command type Memory Read (type 6). Note that during all non-burst read operations, the Am79C974 controller will always activate all byte enables, even though some byte lanes may not contain valid data as indicated by a buffer pointer value. In such instances, the Am79C974 controller will internally discard unneeded bytes.

Figure 6 shows a typical non-burst read access. The Am79C974 controller asserts IRDY at clock 5 immediately after the address phase and starts sampling DEVSEL. The target extends the cycle by asserting DEVSEL not until clock 6. Additionally, the target inserts one wait state by asserting its ready (TRDY) at clock 8.

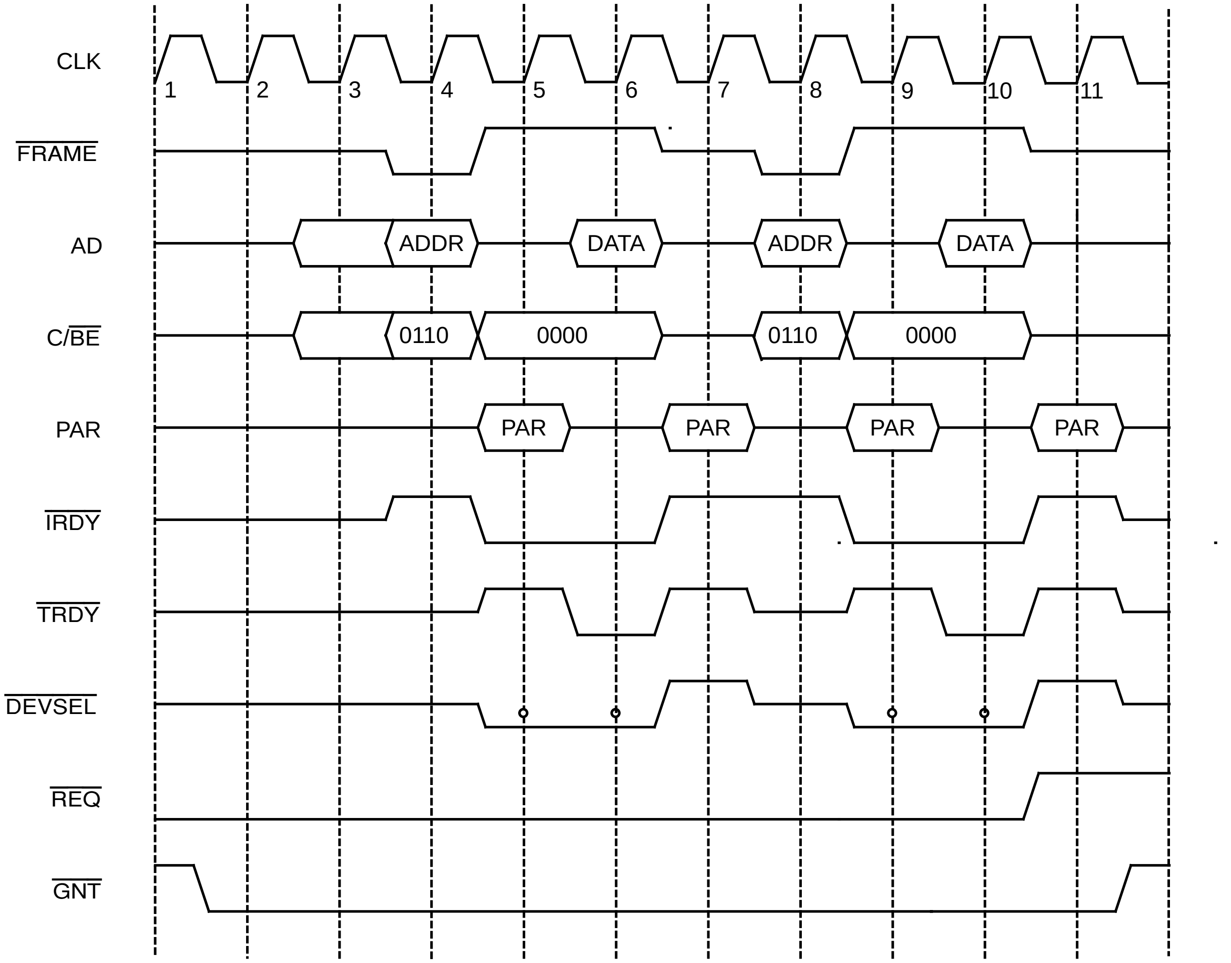


○ DEVSEL *is sampled by the Am79C974 controller.*

18681A-10

**Figure 6. Non-Burst Read Cycles With Wait States**

Figure 7 shows two non-burst read access within one arbitration cycle. The Am79C974 controller will drop FRAME between two consecutive non-burst read cycles. The Am79C974 controller will re-request the bus right again if it is preempted before starting the second access. The example below also shows a target that can respond to the Am79C974 controller read cycles without wait states.



○ DEVSEL *is sampled by the Am79C974 controller.*

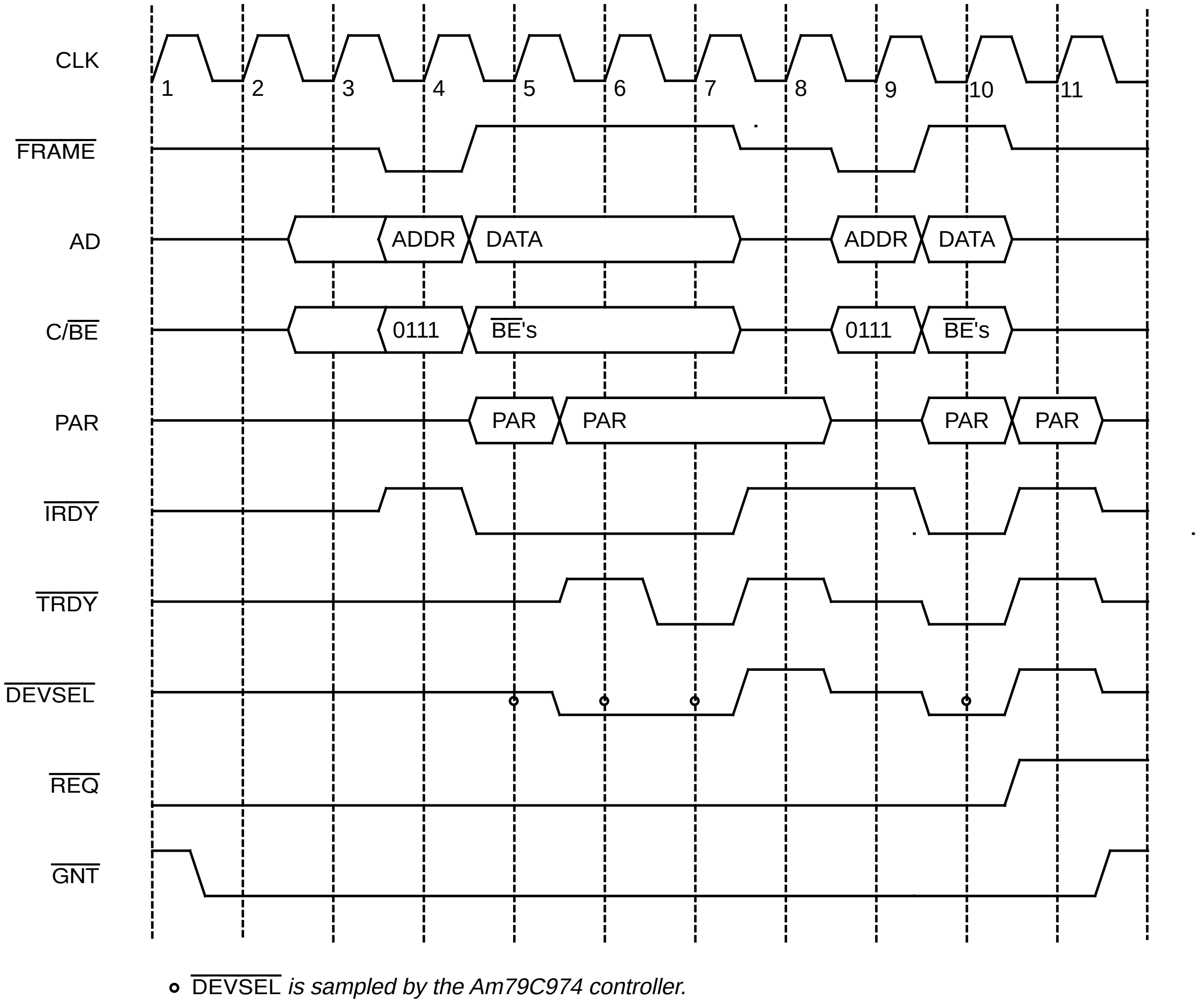
18681A-11

**Figure 7. Non-Burst Read Cycles Without Wait States**

### Basic Non-Burst Write

All Am79C974 controller non-burst write accesses are of the PCI command type Memory Write (type 7).

Figure 8 shows two non-burst write access within one arbitration cycle. The Am79C974 controller will drop FRAME between two consecutive non-burst write cycles. The Am79C974 controller will re-request the bus immediately if it is preempted before starting the second access. The example below shows an extended cycle for the first access. The target asserts DEVSEL 2 clock cycles after the address phase (FRAME asserted) and adds one extra wait state by asserting TRDY only on clock 7. The second write cycle in the example shows a ZERO wait state access.



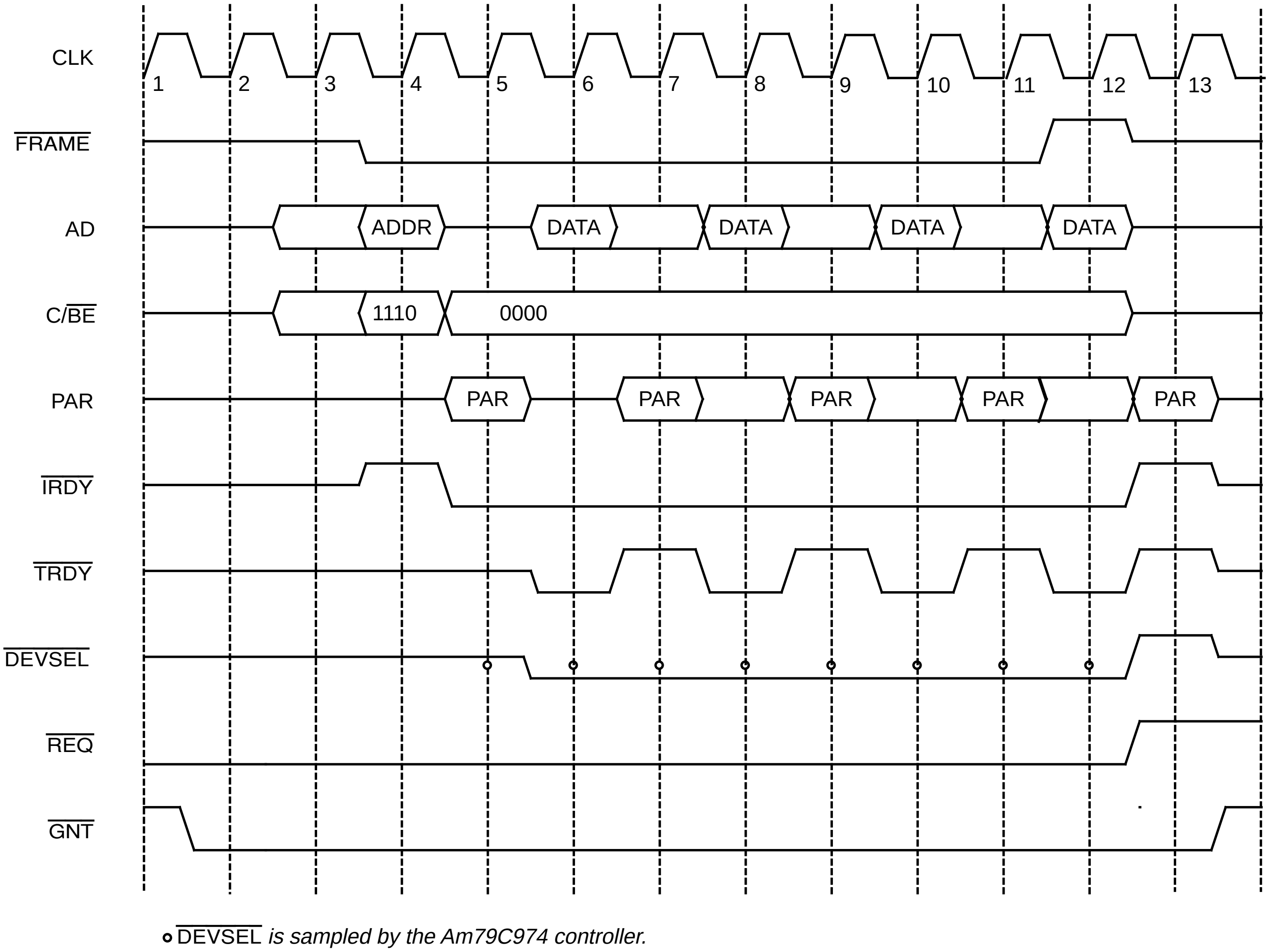
18681A-12

**Figure 8. Non-Burst Write Cycles With and Without Wait States**

### Basic Burst Read Cycles

All Am79C974 controller burst read transfers are of the PCI command type Memory Read Line (type14). AD[1:0] will both be ZERO during the address phase indicating a linear burst order. All four byte enable signals will be ZERO during the data phase as the Am79C974 controller always reads a full 32-bit word when in burst mode.

Figure 9 shows a typical burst read access. The Am79C974 controller arbitrates for the bus, is granted access, and reads four 32-bit words (DWORD) from system memory and then releases the bus. All four data phases in this example take two clock cycles each, which is determined by the timing of TRDY.



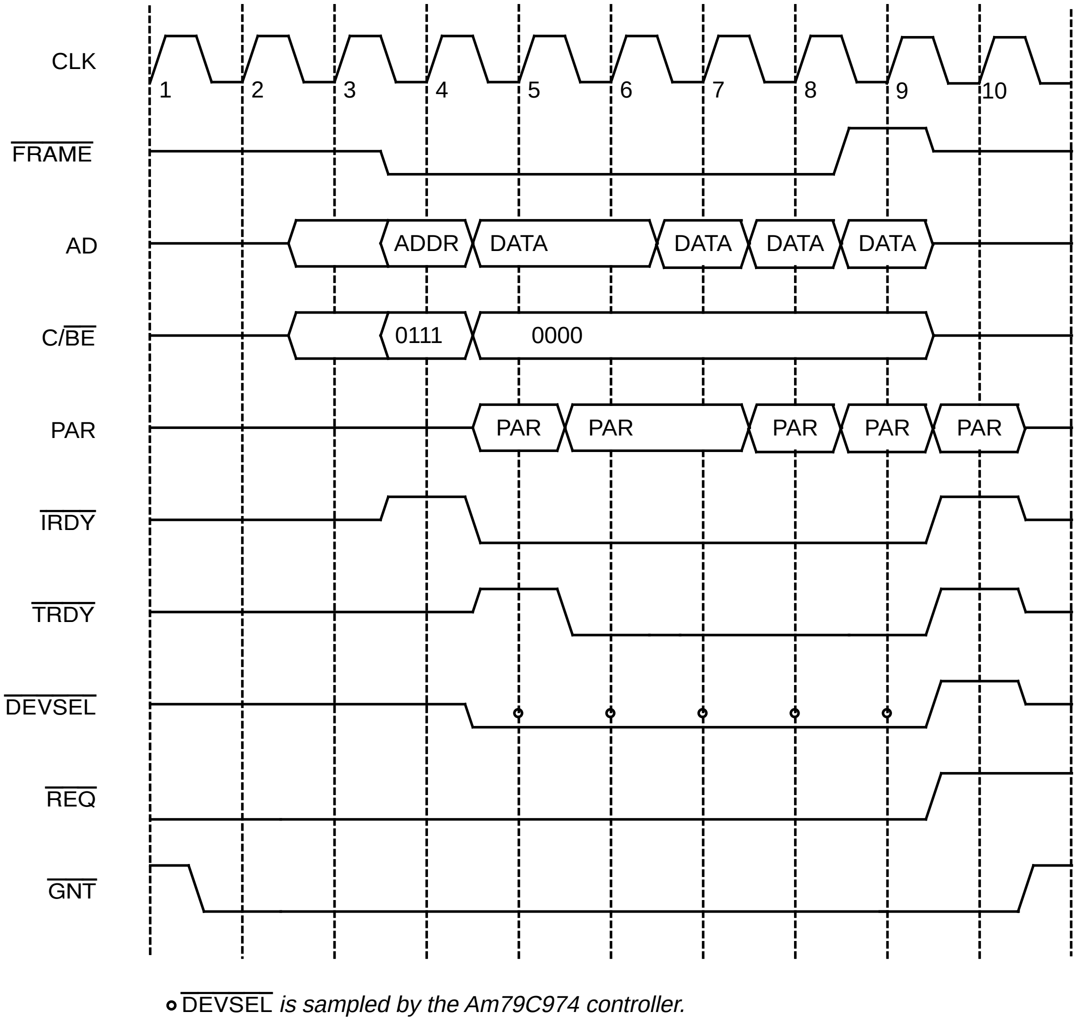
18681A-13

**Figure 9. Burst Read Cycles**

### Basic Burst Write Cycles

All Am79C974 controller burst write transfers are of the PCI command type Memory Write (type 7). AD[1:0] will both be ZERO during the address phase indicating a linear burst order. All four byte enable signals will be ZERO during the data phase as the Am79C974 controller always writes a full 32-bit word when in burst mode.

Figure 10 shows a typical burst write access. The Am79C974 controller arbitrates for the bus, is granted access, and writes four 32-bit words (DWORDs) from system memory and then releases the bus. In this example, the memory system extends the data phase of the first access by one wait state. The following three data phases take one clock cycle each, which is determined by the timing of $\overline{\text{TRDY}}$.



18681A-14

**Figure 10. Burst Write Cycles**

## Transaction Termination

Termination of a PCI transaction may be initiated by either the master or the target. During termination, the master remains in control to bring all PCI transactions to an orderly and systematic conclusion regardless of what caused the termination. All transactions are concluded when $\overline{\text{FRAME}}$ and $\overline{\text{IRDY}}$ are both deasserted, indicating an IDLE cycle.

### Target Initiated Termination

When the Am79C974 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 11 shows a disconnection in which one last data transfer occurs after the target asserted $\overline{\text{STOP}}$. $\overline{\text{STOP}}$ is asserted on clock 4 to start the termination sequence. Data is still transferred during this cycles, since both $\overline{\text{IRDY}}$ and $\overline{\text{TRDY}}$ are asserted. The Am79C974 controller terminates the current transfer with the deassertion of $\overline{\text{FRAME}}$ on clock 5 and then one clock cycle later with the deassertion of $\overline{\text{IRDY}}$. It finally releases the bus on clock 6. The Am79C974 controller will re-request the bus after 2 clock cycles, if it wants to transfer more data. The starting address of the new transfer will be the address of the next untransferred data.

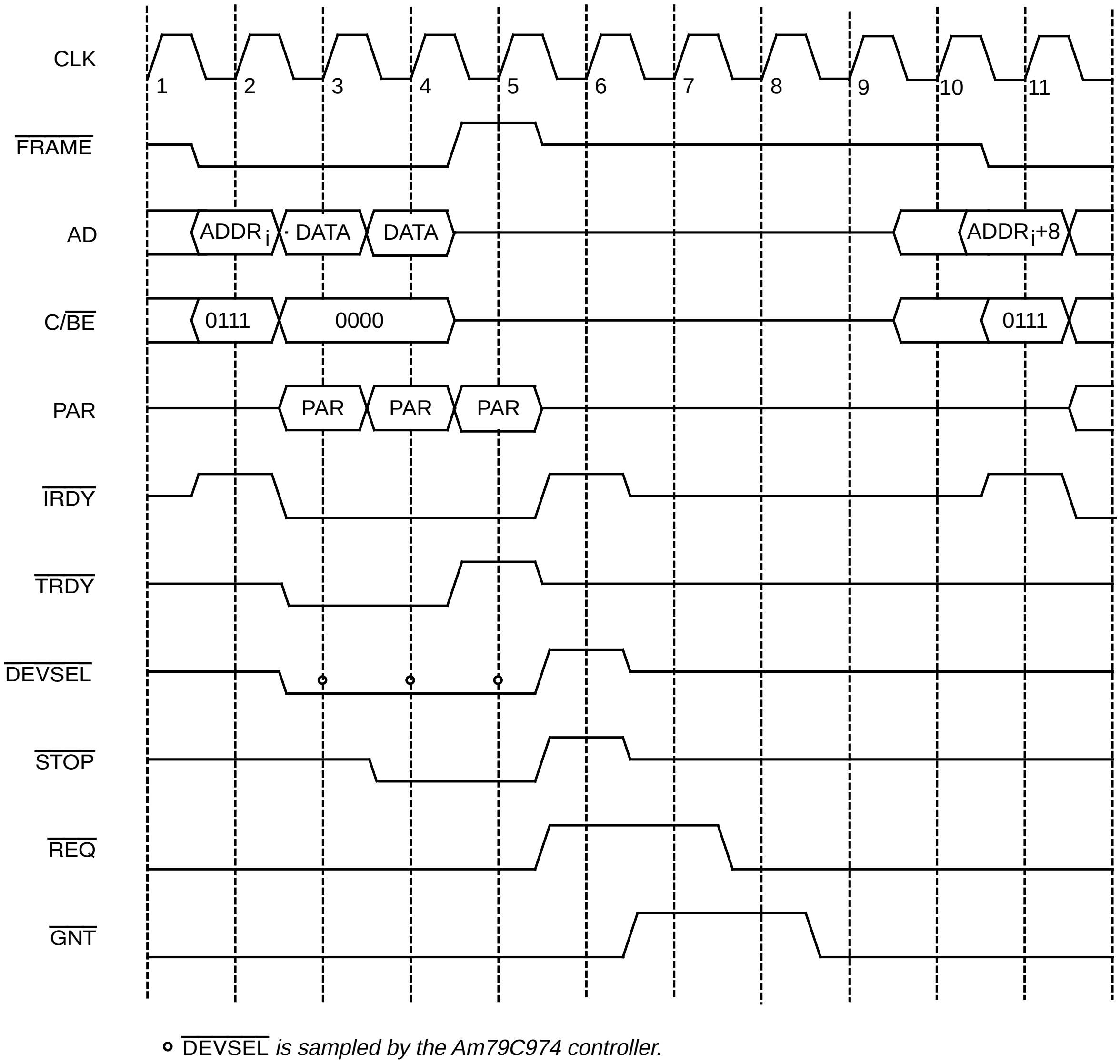
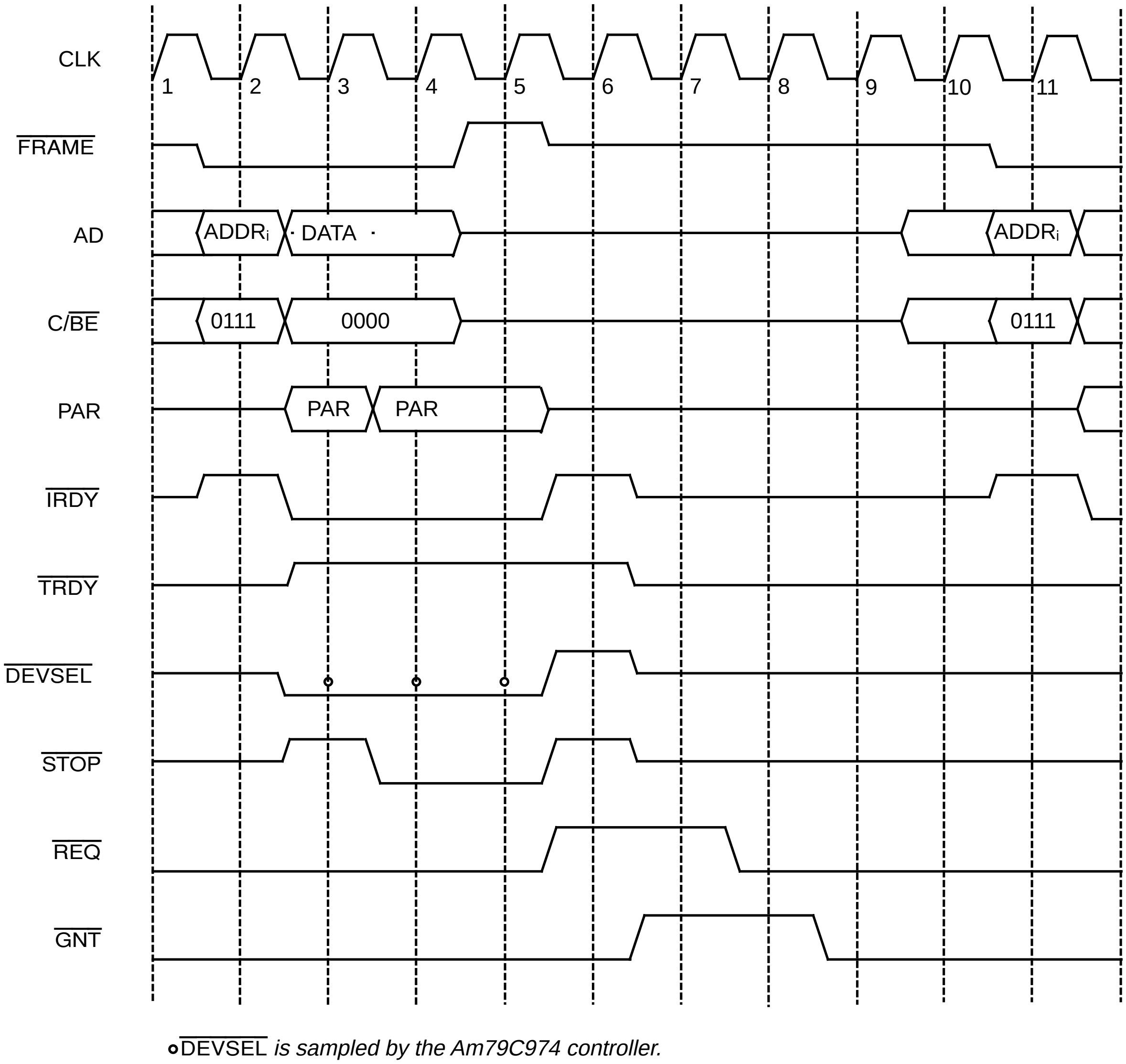


**Figure 11. Disconnect with Data Transfer**

### Disconnect Without Data Transfer

Figure 12 shows a target disconnect sequence during which no data is transferred. $\overline{\text{STOP}}$ is asserted on clock 4 without $\overline{\text{TRDY}}$ being asserted at the same time. The Am79C974 controller terminates the current transfer with the deassertion of $\overline{\text{FRAME}}$ on clock 5 and one clock cycle later with the deassertion of $\overline{\text{IRDY}}$. It finally releases the bus on clock 6. The Am79C974 controller will re-request the bus after 2 clock cycles to retry the last transfer. The starting address of the new transfer will be the same address as of the last untransferred data.



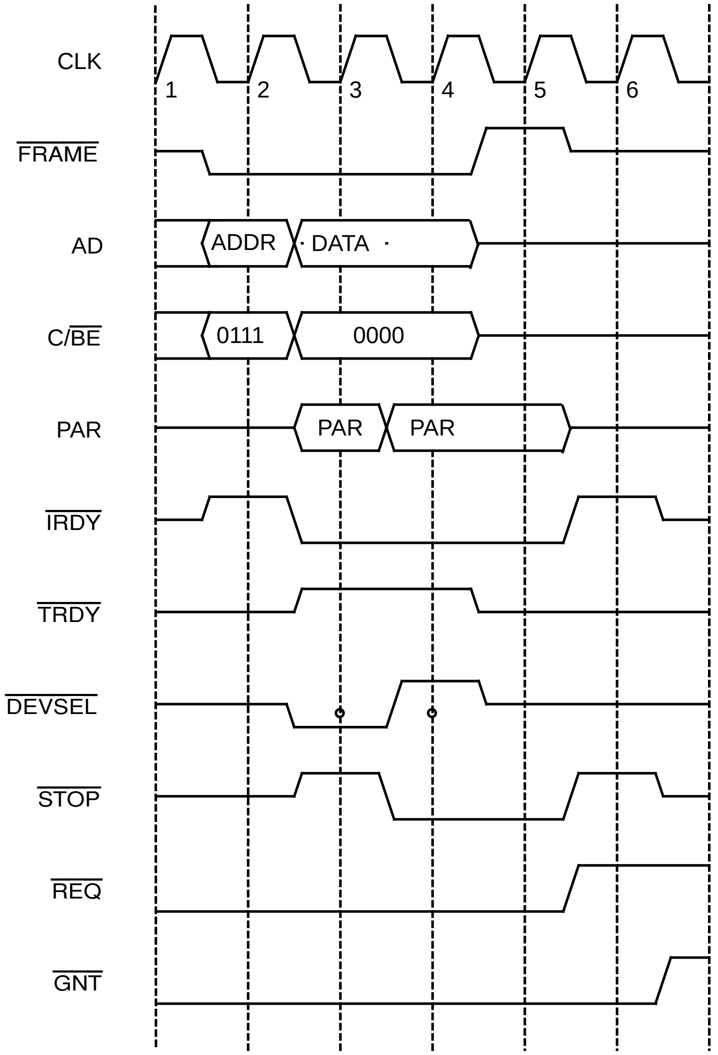
18681A-16

**Figure 12. Disconnect Without Data Transfer**

### Target Abort

Figure 13 shows a target abort sequence. The target asserts $\overline{\text{DEVSEL}}$ for one clock. It then deasserts $\overline{\text{DEVSEL}}$ and asserts $\overline{\text{STOP}}$ 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 Am79C974 controller cannot make any assumption about the success of the previous data transfers in the current transaction. The Am79C974 controller terminates the current transfer with the deassertion of $\overline{\text{FRAME}}$ on clock 5 and one clock cycle later with the deassertion of $\overline{\text{IRDY}}$. It finally releases the bus on clock 6.

Since data integrity is not guaranteed, the Am79C974 controller cannot recover from a target abort event. For Ethernet, the Am79C974 controller will reset all CSR and BCR locations to their H_RESET values. Any ongoing network activity will be stopped immediately. The PCI configuration registers will not be cleared. For SCSI, when target aborts, $\overline{\text{INTA}}$ will not be asserted, but the ABORT bit (bit 2 of the DMA status register at offset 54h), is set. For either Ethernet or SCSI a target abort causes RTABORT (bit 12) of the status register in the appropriate PCI configuration space to be set.



○ $\overline{\text{DEVSEL}}$ *is sampled by the Am79C974 controller.*

18681A-17

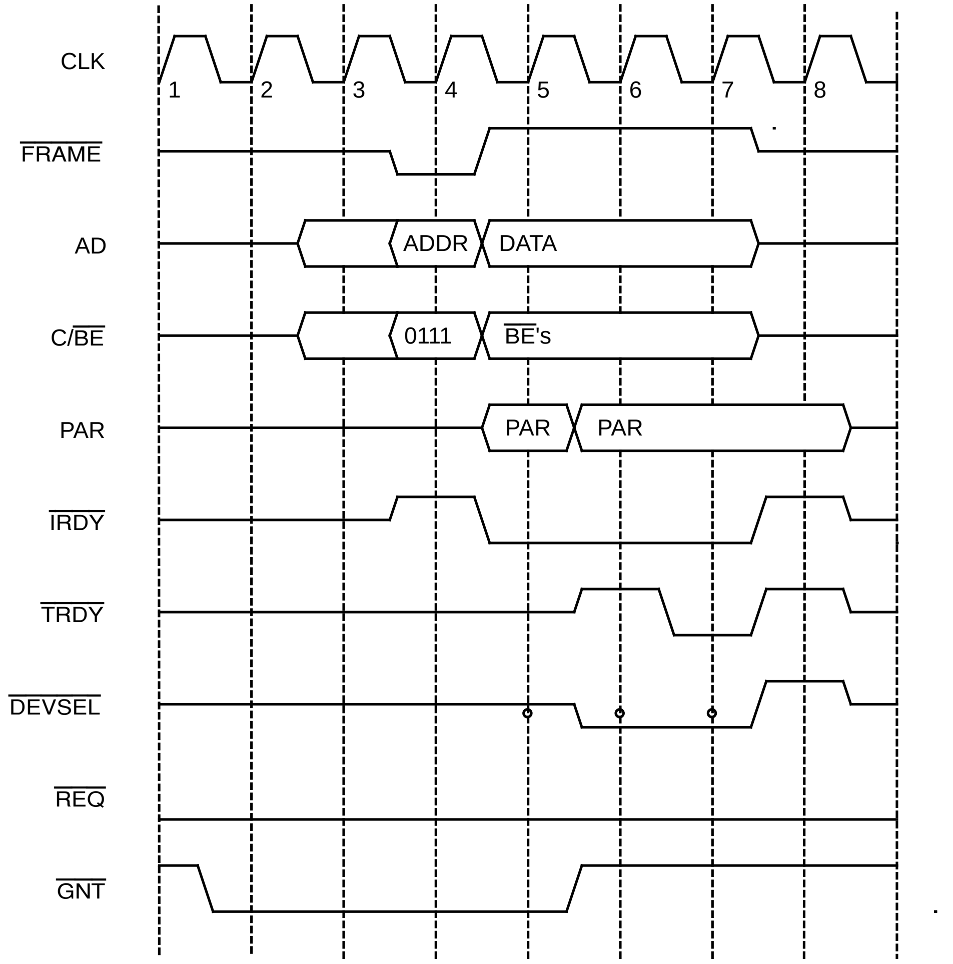
**Figure 13. Target Abort**

### Master Initiated Termination

There are three scenarios besides normal completion of a transaction where the Am79C974 controller will terminate the cycles it produces on the PCI bus. These are Preemption with and without FRAME assertion and Master Abort.

### Preemption When FRAME is Deasserted

The Am79C974 controller can generate multiple address phases during a single bus ownership period when transferring data. FRAME is deasserted in between two address phases. While FRAME is deasserted, the central arbiter can remove GNT to the Am79C974 controller at any time to service another master. When GNT is removed, the Am79C974 controller will finish the current transfer and then release the bus. It will keep REQ asserted to regain bus ownership as soon as possible.



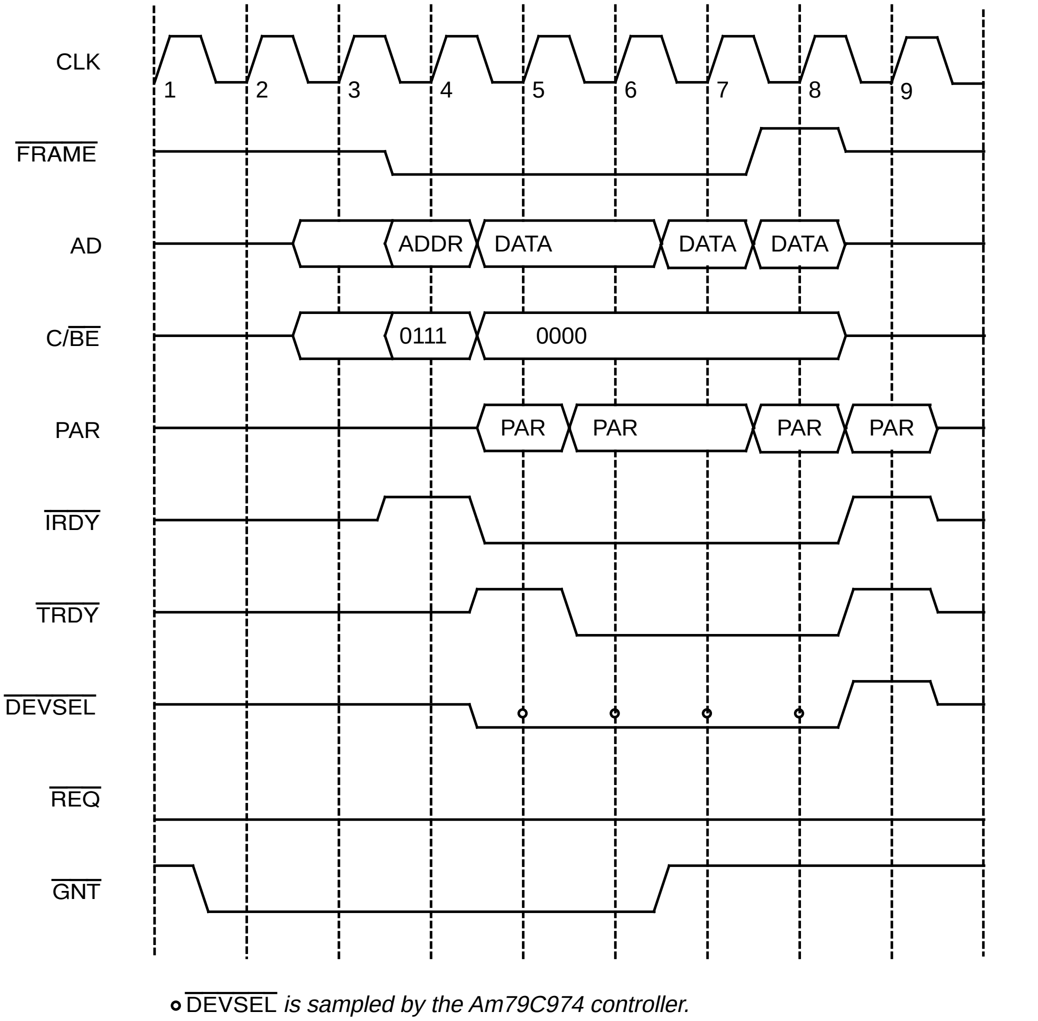
○ DEVSEL *is sampled by the Am79C974 controller.*

18681A-18

**Figure 14. Preemption When FRAME is Deasserted**

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

The central arbiter can take $\overline{\text{GNT}}$ to the Am79C974 controller away if the current bus operation takes too long. This may happen, for example, when the Am79C974 controller tries to fill the whole Ethernet transmit FIFO and the target inserts extra wait states for every data phase. When $\overline{\text{GNT}}$ is taken away, the Am79C974 controller will finish the current transfer and then immediately release the bus. The Latency Timer in PCI configuration space of the Am79C974 controller is always set to ZERO. The Am79C974 controller will keep $\overline{\text{REQ}}$ asserted to regain bus ownership as soon as possible.

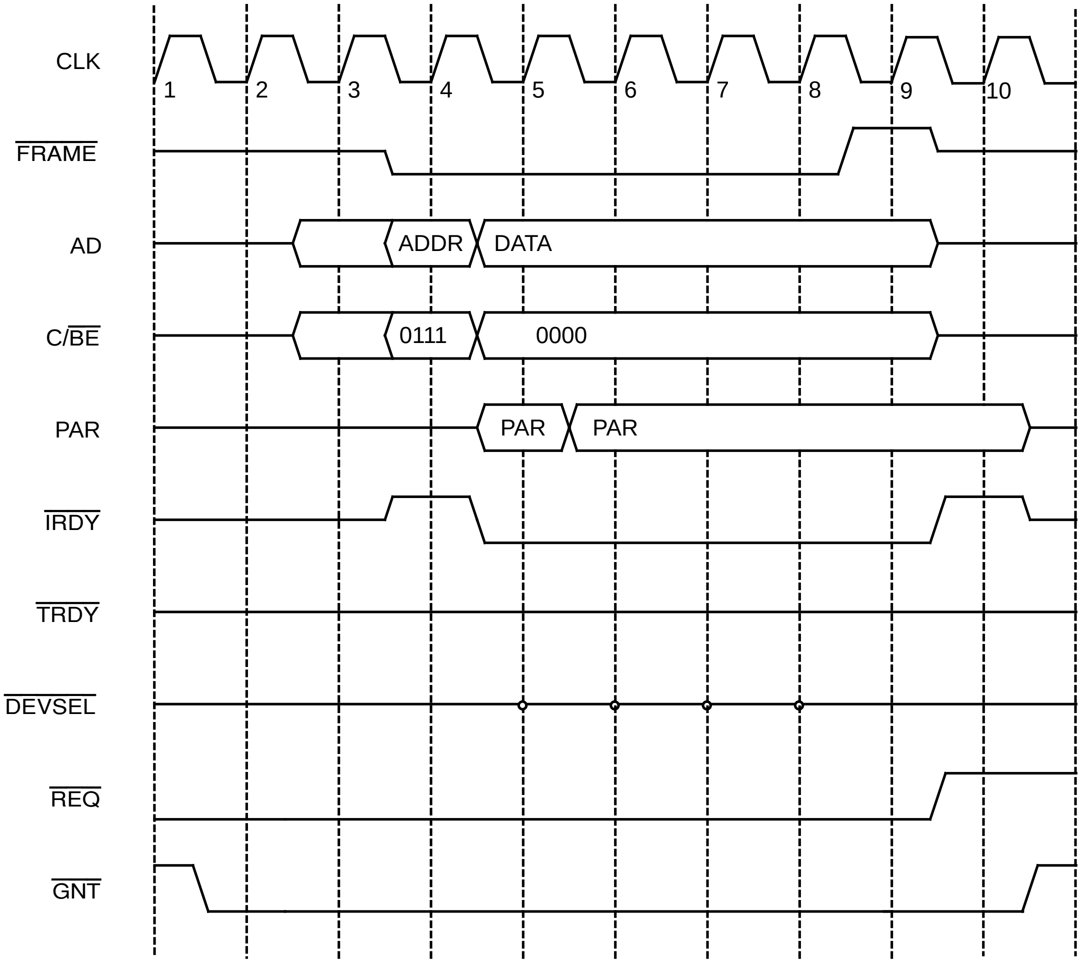


18681A-19

**Figure 15. Preemption When $\overline{\text{FRAME}}$ is Asserted**

### Master Abort

The Am79C974 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 Am79C974 controller. For the Ethernet, the Am79C974 controller will reset all CSR and BCR locations to their H_RESET values. Any on-going network activity will be stopped immediately. The PCI configuration registers will not be cleared. For SCSI, when the master aborts, INTA will not be asserted, but the ABORT bit (bit 2 of the DMA status register at offset 54h), is set. For either Ethernet or SCSI master abort causes RMABORT (bit 13) of the status register in the appropriate PCI configuration space to be set.



○ DEVSEL *is sampled by the Am79C974 controller.*

18681A-20

**Figure 16. Master Abort**

## Ethernet Controller

### Buffer Management Unit (BMU)

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

#### *Initialization*

Am79C974 initialization includes the reading of the initialization block in memory to obtain the operating parameters. The initialization block is read when the INIT bit in CSR0 is set. The INIT bit should be set before or concurrent with the STRT bit to insure correct operation. Two DWORDs are read during each period of bus mastership. When SSIZE32 = 1 (BCR20, bit 8), this results in a total of 4 arbitration cycles (3 arbitration cycles if SSIZE32 = 0). Once the initialization block has been completely read in and internal registers have been updated, IDON will be set in CSR0, and an interrupt generated (if IENA is set). At this point, the BMU knows where the receive and transmit descriptor rings and hence, normal network operations will begin.

The Am79C974 controller obtains the start address of the Initialization Block from the contents of CSR1 (least significant 16 bits of address) and CSR2 (most significant 16 bits of address). The host must write CSR1 and CSR2 before setting the INIT bit. The block contains the user defined conditions for Am79C974 operation, together with the base addresses and length information of the transmit and receive descriptor rings.

There is an alternative method to initialize the Am79C974 controller. Instead of initialization via the initialization block in memory, data can be written directly into the appropriate registers. Either method may be used at the discretion of the programmer. If the registers are written to directly, the INIT bit must not be set, or the initialization block will be read in, thus overwriting the previously written information. Please refer to Appendix C for details on this alternative method.

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

#### *Re-Initialization*

The transmitter and receiver sections of the Am79C974 controller can be turned on via the initialization block (MODE Register DTX, DRX bits; CSR15[1:0]). The states of the transmitter and receiver are monitored by the host through CSR0 (RXON, TXON bits). The Am79C974 controller should be reinitialized if the transmitter and/or the receiver were not turned on during the original initialization, and it was subsequently required to activate them or if either section was shut off due to the detection of certain error conditions (MERR, UFLO, TX BUFF error).

Reinitialization may be done via the initialization block or by setting the STOP bit in CSR0, followed by writing to CSR15, and then setting the START bit in CSR0. Note that this form of restart will not perform the same in the Am79C974 controller as in the LANCE. In particular, upon restart, the Am79C974 controller reloads the transmit and receive descriptor pointers with their respective base addresses. This means that the software must clear the descriptor own bits and reset its descriptor ring pointers before the restart of the Am79C974 controller. The reload of descriptor base addresses is performed in the LANCE only after initialization, so a restart of the LANCE without initialization leaves the LANCE pointing at the same descriptor locations as before the restart.

#### *Buffer Management*

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

#### *Descriptor Rings*

Each descriptor ring must be organized in a contiguous area of memory. At initialization time (setting the INIT bit in CSR0), the Am79C974 controller reads the user-defined base address for the transmit and receive descriptor rings, as well as the number of entries contained in the descriptor rings. Descriptor ring base addresses must be on a 16-byte boundary when SSIZE32=1, and 8-byte boundary when SSIZE=0. A maximum of 128 (or 512, depending upon the value of SSIZE32) ring entries is allowed when the ring length is set through the TLEN and RLEN fields of the initialization block. However, the ring lengths can be set beyond this range (up to 65535) by writing the transmit and receive ring length registers (CSR76, CSR78) directly.

Each ring entry contains the following information:

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

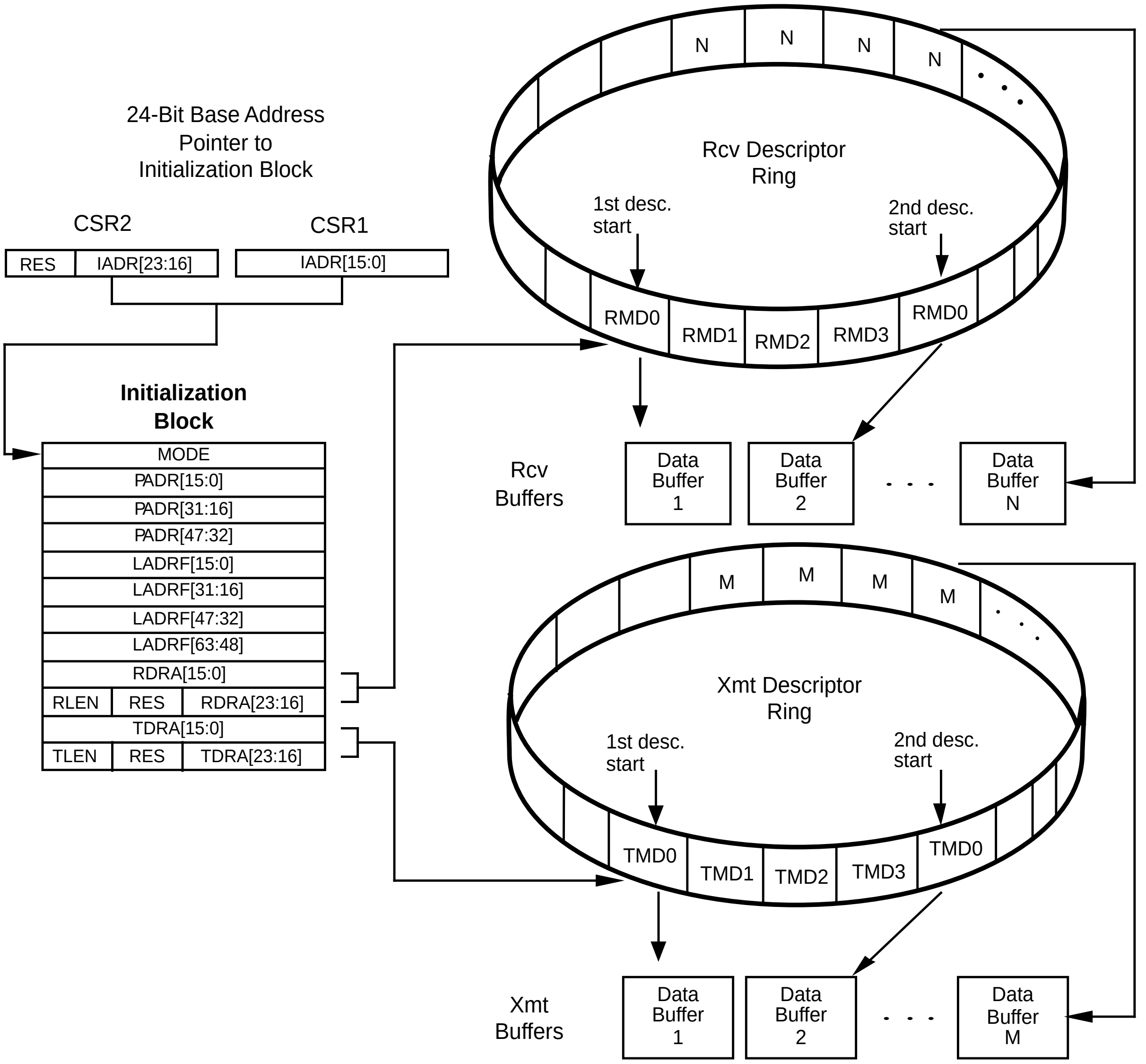
To permit the queuing and de-queuing of message buffers, ownership of each buffer is allocated to either the Am79C974 controller or the host. The OWN bit within the descriptor status information, either TMD or RMD (see section on TMD or RMD), is used for this purpose. OWN = “1” signifies that the Am79C974 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 Am79C974 controller, then the software must not read "ahead" to the next descriptor. The software should wait at the unOWNed descriptor until ownership has been granted to the software (when LAPPEN = 1 (CSR3, bit 5), then this rule is modified. See the LAPPEN description). Strict adherence to these rules insures that "Deadly Embrace" conditions are avoided.

### *Descriptor Ring Access Mechanism*

At initialization, the Am79C974 controller reads the base address of both the transmit and receive descriptor rings into CSRs for use by the Am79C974 controller during subsequent operations.

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

When SSIZE32 = 0, software data structures are 16 bits wide. The following diagram, Figure 17, illustrates the relationship between the Initialization Base Address, the Initialization Block, the Receive and Transmit Descriptor Ring Base Addresses, the Receive and Transmit Descriptors and the Receive and Transmit Data Buffers, for the case of SSIZE32 = 0.



18681A-21

**Figure 17. 16-Bit Data Structures: Initialization Block and Descriptor Rings**

When SSIZE32 = 1, software data structures are 32 bits wide. The following diagram illustrates, Figure 18, the relationship between the Initialization Base Address, the Initialization Block, the Receive and Transmit Descriptor Ring Base Addresses, the Receive and Transmit Descriptors and the Receive and Transmit Data Buffers, for the case of SSIZE32 = 1.

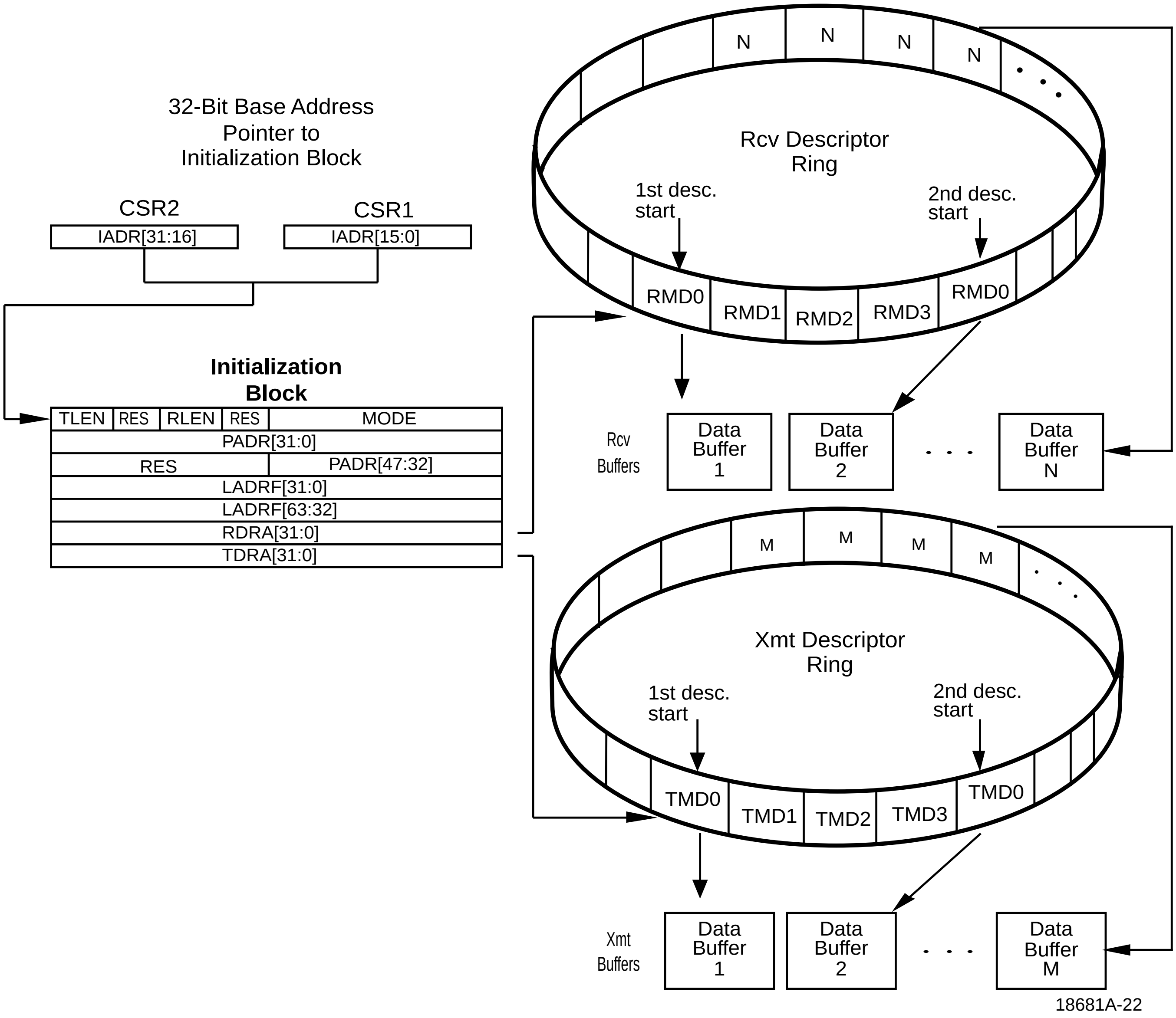


**Figure 18. 32-Bit Data Structures: Initialization Block and Descriptor Rings**

### *Polling*

If there is no network channel activity and there is no pre- or post-receive or pre- or post-transmit activity being performed by the Am79C974 controller, then the Am79C974 controller will periodically poll the current receive and transmit descriptor entries in order to ascertain their ownership. If the DPOLL bit in CSR4 is set, then the transmit polling function is disabled.

A typical polling operation consists of the following: The Am79C974 controller will use the current receive descriptor address stored internally to vector to the appropriate Receive Descriptor Table Entry (RDTE). It will then use the current transmit descriptor address (stored internally) to vector to the appropriate Transmit Descriptor Table Entry (TDTE). The accesses will be made in the following order: RMD1, then RMD0 of the current RDTE during one bus arbitration, and after that, TMD1, then TMD0 of the current TDTE during a second bus arbitration. All information collected during polling activity will be stored internally in the appropriate CSRs. (i.e. CSR18, CSR19, CSR20, CSR21, CSR40, CSR42, CSR50, CSR52). UnOWNed descriptor status will be internally ignored.

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

1. Am79C974 controller does not possess ownership of the current RDTE and the poll time has elapsed and RXON=1 (CSR0, bit 5), or
2. Am79C974 controller does not possess ownership of the next RDTE the poll time has elapsed and RXON=1.

If RXON=0 the Am79C974 controller will never poll RDTE locations.

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

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

1. Am79C974 controller does not possess ownership of the current TDTE and
   DPOLL=0 (CSR4, bit 2) and
   TXON=1 (CSR0, bit 4) and
   the poll time has elapsed, or
2. Am79C974 controller does not possess ownership of the current TDTE and
   DPOLL=0 and
   TXON=1 and
   a frame has just been received, or
3. Am79C974 controller does not possess ownership of the current TDTE and
   DPOLL=0 and
   TXON=1 and
   a frame has just been transmitted.

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

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

### *Transmit Descriptor Table Entry (TDTE)*

If, after a TDTE access, the Am79C974 controller finds that the OWN bit of that TDTE is not set, then the Am79C974 controller resumes the poll time count and reexamines the same TDTE at the next expiration of the poll time count.

If the OWN bit of the TDTE is set, but Start of Frame (STP) bit is not set, the Am79C974 controller will immediately request the bus in order to reset the OWN bit of this descriptor. (This condition would normally be found following a LCOL or RETRY error that occurred in the middle of a transmit frame chain of buffers.) After resetting the OWN bit of this descriptor, the Am79C974 controller will again immediately request the bus in order to access the next TDTE location in the ring.

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

If the OWN bit is set and the start of frame (STP) bit is set, then microcode control proceeds to a routine that will enable transmit data transfers to the FIFO. The

Am79C974 controller will look ahead to the next transmit descriptor after it has performed at least one transmit data transfer from the first buffer. (More than one transmit data transfer may possibly take place, depending upon the state of the transmitter.) The contents of TMD0 and TMD1 will be stored in Next Xmt Buffer Address (CSR64 and CSR65), Next Xmt Byte Count (CSR66) and Next Xmt Status (CSR67) regardless of the state of the OWN bit. This transmit descriptor lookahead operation is performed only once.

If the Am79C974 controller does not own the next TDTE (i.e. the second TDTE for this frame), then it will complete transmission of the current buffer and then update the status of the current (first) TDTE with the BUFF and UFLO bits being set. This will cause the transmitter to be disabled (CSR0, TXON=0). The Am79C974 controller will have to be re-initialized to restore the transmit function. The situation that matches this description implies that the system has not been able to stay ahead of the Am79C974 controller in the transmit descriptor ring and therefore, the condition is treated as a fatal error. (To avoid this situation, the system should always set the transmit chain descriptor own bits in reverse order.)

If the Am79C974 controller does own the second TDTE in a chain, it will gradually empty the contents of the first buffer (as the bytes are needed by the transmit operation), perform a single-cycle DMA transfer to update the status of the first descriptor (reset the OWN bit in TMD1), and then it may perform one data DMA access on the second buffer in the chain before executing another lookahead operation. (i.e. a lookahead to the third descriptor.)

The Am79C974 controller can queue up to two frames in the transmit FIFO. Call them frame “X” and frame “Y”, where “Y” is after “X”. Assume that frame “X” is currently being transmitted. Because the Am79C974 controller can perform lookahead data transfer past the ENP of frame “X”, it is possible for the Am79C974 controller to completely transfer the data from a buffer belonging to frame “Y” into the FIFO even though frame “X” has not yet been completely transmitted. At the end of this “Y” buffer data transfer, the Am79C974 controller will write intermediate status (change the OWN bit to a ZERO) for the “Y” frame buffer, if frame “Y” uses data chaining. The last TDTE for the “X” frame (containing ENP) has not yet been written, since the “X” frame has not yet been completely transmitted. Note that the Am79C974 controller has, in this instance, returned ownership of a TDTE to the host out of a “normal” sequence.

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

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

If an error occurs in the transmission before all of the bytes of the current buffer have been transferred, then TMD2 and TMD1 of the current buffer will be written; In such a case, data transfers from the next buffer will not commence. Instead, following the TMD2/TMD1 update, the Am79C974 controller will go to the next transmit frame, if any, skipping over the rest of the frame which experienced an error, including chained buffers. This is done by returning to the polling microcode where Am79C974 controller will immediately access the next descriptor and find the condition OWN=1 and STP=0 as described earlier. As described for that case, the Am79C974 controller will reset the own bit for this descriptor and continue in like manner until a descriptor with OWN=0 (no more transmit frames in the ring) or OWN=1 and STP=1 (the first buffer of a new frame) is reached.

At the end of any transmit operation, whether successful or with errors, immediately following the completion of the descriptor updates, the Am79C974 controller will always perform another poll operation. As described earlier, this poll operation will begin with a check of the current RDTE, unless the Am79C974 controller already owns that descriptor. Then the Am79C974 controller will proceed to polling the next TDTE. If the transmit descriptor OWN bit has a ZERO value, then the Am79C974 controller will resume poll time count incrementing. If the transmit descriptor OWN bit has a value of ONE, then the Am79C974 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 Am79C974 controller to avoid inserting poll time counts between successive transmit frames.

Whenever the Am79C974 controller completes a transmit frame (either with or without error) and writes the status information to the current descriptor, then the TINT bit of CSR0 is set to indicate the completion of a transmission. This causes an interrupt signal if the IENA bit of CSR0 has been set and the TINTM bit of CSR3 is reset.

***Receive Descriptor Table Entry (RDTE)***

If the Am79C974 controller does not own both the current and the next Receive Descriptor Table Entry then the Am79C974 controller will continue to poll according to the polling sequence described above. If the receive descriptor ring length is 1, then there is no next descriptor to be polled.

If a poll operation has revealed that the current and the next RDTE belong to the Am79C974 controller then additional poll accesses are not necessary. Future poll operations will not include RDTE accesses as long as the Am79C974 controller retains ownership of the current and the next RDTE.

When receive activity is present on the channel, the Am79C974 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 Am79C974 controller checks the current receive buffer status register CRST (CSR41) to determine the ownership of the current buffer.

If ownership is lacking, then the Am79C974 controller will immediately perform a (last ditch) poll of the current RDTE. If ownership is still denied, then the Am79C974 controller has no buffer in which to store the incoming message. The Missed Frame Count register (CSR112) will be incremented and the MISS bit will be set in CSR0 and an interrupt will be generated if IENA=1 (CSR0) and MISSM=0 (CSR3). Another poll of the current RDTE will not occur until the frame has finished.

If the Am79C974 controller sees that the last poll (either a normal poll, or the last-ditch effort described in the above paragraph) of the current RDTE shows valid ownership, then it proceeds to a poll of the next RDTE. Following this poll, and regardless of the outcome of this poll, transfers of receive data from the FIFO may begin.

Regardless of ownership of the second receive descriptor, the Am79C974 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 Am79C974 controller does not own the second buffer, ownership of the current descriptor will be passed back to the system by writing a ZERO to the OWN bit of RMD1 and status will be written indicating buffer (BUFF=1) and possibly overflow (OFLO=1) errors.

If the frame length exceeds the length of the first (current) buffer, and the Am79C974 controller does own the second (next) buffer, ownership will be passed back to the system by writing a ZERO to the OWN bit of RMD1 when the first buffer is full. Receive data transfers to the second buffer may occur before the Am79C974 controller proceeds to look ahead to the ownership of the third buffer. Such action will depend upon the state of the FIFO when the status has been updated on the first descriptor. In any case, lookahead will be performed to the third buffer and the information gathered will be stored in the chip, regardless of the state of the ownership bit. As in the transmit flow, lookahead operations are performed only once.

This activity continues until the Am79C974 controller recognizes the completion of the frame (the last byte of this receive message has been removed from the FIFO). The Am79C974 controller will subsequently update the current RDTE status with the end of frame (ENP) indication set, write the message byte count (MCNT) of the complete frame into RMD2 and overwrite the “current” entries in the CSRs with the “next” entries.

## Media Access Control

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

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

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

The two primary attributes of the MAC engine are:

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

### Transmit and Receive Message Data Encapsulation

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

ASTRP_RCV = 1 (CSR4, bit 10), the receiver will automatically strip pad bytes from the received message by observing the value in the length field, and stripping excess bytes if this value is below the minimum data size (46 bytes). Both features can be independently over-ridden to allow illegally short (less than 64 bytes of frame data) messages to be transmitted and/or received. The use of this feature reduces bus utilization because the pad bytes are not transferred into or out of main memory.

### Framing (Frame Boundary Delimitation, Frame Synchronization)

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

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

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

If the frame terminates or suffers a collision before 64-bytes of information (after SFD) have been received, the MAC engine will automatically delete the frame from the Receive FIFO, without host intervention. The Am79C974 controller has the ability to accept runt packets for diagnostics purposes and proprietary networks.

### Addressing (Source and Destination Address Handling)

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

### Error Detection (Physical Medium Transmission Errors)

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

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

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

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

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

The Am79C974 controller can handle up to 7 dribbling bits when a received frame terminates. During the reception, the FCS is generated on every serial bit (including the dribbling bits) coming from the cable, although the internally saved FCS value is only updated on the eighth bit (on each byte boundary). The framing error is reported to the user as follows:

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

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

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

### Media Access Management

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

### Medium Allocation

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

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

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

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

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

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

The Am79C974 controller will perform the two part deferral algorithm as specified in Section 4.2.8 (Process Deference). The Inter Packet Gap (IPG) timer will start timing the 9.6 µs InterFrameSpacing after the receive carrier is de-asserted. During the first part deferral (InterFrameSpacingPart1 – IFS1) the Am79C974 controller will defer any pending transmit frame and respond to the receive message. The IPG counter will be reset to ZERO continuously until the carrier de-asserts, at which point the IPG counter will resume the 9.6 µs count once again. Once the IFS1 period of 6.0 µs has elapsed, the Am79C974 controller will begin timing the second part deferral (InterFrame Spacing Part 2 – IFS2) of 3.6 µs. Once IFS1 has completed, and IFS2 has commenced, the Am79C974 controller will not defer to a receive frame if a transmit frame is pending. This means that the Am79C974 controller will not attempt to receive the receive frame, since it will start to transmit, and generate a collision at 9.6 µs. The Am79C974 controller will guarantee to complete the preamble (64-bit) and jam (32-bit) sequence before ceasing transmission and invoking the random backoff algorithm.

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

During the time period immediately after a transmission has been completed, the external transceiver (in the case of a standard AUI connected device), should generate the SQE Test message (a nominal 10 MHz burst of 5 –15 Bit Times duration) on the CI± pair (within 0.6 µs – 1.6 µs after the transmission ceases). During the time period in which the SQE Test message is expected the Am79C974 controller will not respond to receive carrier sense.

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

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

The Am79C974 controller implements a carrier sense “blinding” period within 0 µs – 4.0 µs from de-assertion of carrier sense after transmission. This effectively means that when transmit two part deferral is enabled (DXMT2PD is cleared) the IFS1 time is from 4 µs to 6 µs after a transmission. However, since IPG shrinkage below 4 µs will rarely be encountered on a correctly configured networks, and since the fragment size will be larger than the 4 µs blinding window, then the IPG counter will be reset by a worst case IPG shrinkage/fragment scenario and the Am79C974 controller will defer its transmission. In addition, the Am79C974 controller will not restart the “blinding” period if carrier is detected within the 4.0 µs – 6.0 µs IFS1 period, but will commence timing of the entire IFS1 period.

### Contention Resolution (Collision Handling)

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

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

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

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

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

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

0 £ r <2k

where

k = min (n,10).”

The Am79C974 controller provides an alternative algorithm, which suspends the counting of the slot time/IPG during the time that receive carrier sense is detected. This aids in networks where large numbers of nodes are present, and numerous nodes can be in collision. It effectively accelerates the increase in the backoff time in busy networks, and allows nodes not involved in the collision to access the channel whilst the colliding nodes await a reduction in channel activity. Once channel activity is reduced, the nodes resolving the collision time out their slot time counters as normal.

## Manchester Encoder/Decoder (MENDEC)

The integrated Manchester Encoder/Decoder provides the PLS (Physical Layer Signaling) functions required for a fully compliant ISO 8802-3 (IEEE/ANSI 802.3) station. The MENDEC provides the encoding function for data to be transmitted on the network using the high accuracy on-board oscillator, driven by either the crystal oscillator or an external CMOS level compatible clock. The MENDEC also provides the decoding function from data received from the network. The MENDEC contains a Power On Reset (POR) circuit, which ensures that all analog portions of the Am79C974 controller are forced into their correct state during power up, and prevents erroneous data transmission and/or reception during this time.

### External Crystal Characteristics

When using a crystal to drive the oscillator, the following crystal specification may be used to ensure less than ±0.5 ns jitter at DO±. See Table 1 below.

**Table 1. Crystal Specification**

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

### External Clock Drive Characteristics

When driving the oscillator from a CMOS level external clock source, XTAL2 must be left floating (unconnected). An external clock having the following characteristics must be used to ensure less than ±0.5 ns jitter at DO±. See Table 2.

**Table 2. Clock Drive Characteristics**

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

### MENDEC Transmit Path

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

### Transmitter Timing and Operation

A 20 MHz fundamental mode crystal oscillator provides the basic timing reference for the MENDEC portion of the Am79C974 controller. The crystal is divided by two, to create the internal transmit clock reference. Both clocks are fed into the MENDECs Manchester Encoder to generate the transitions in the encoded data stream. The internal transmit clock is used by the MENDEC to internally synchronize the Internal Transmit Data (ITXDAT) from the controller and Internal Transmit Enable (ITXEN). The internal transmit clock is also used as a stable bit rate clock by the receive section of the MENDEC and controller.

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

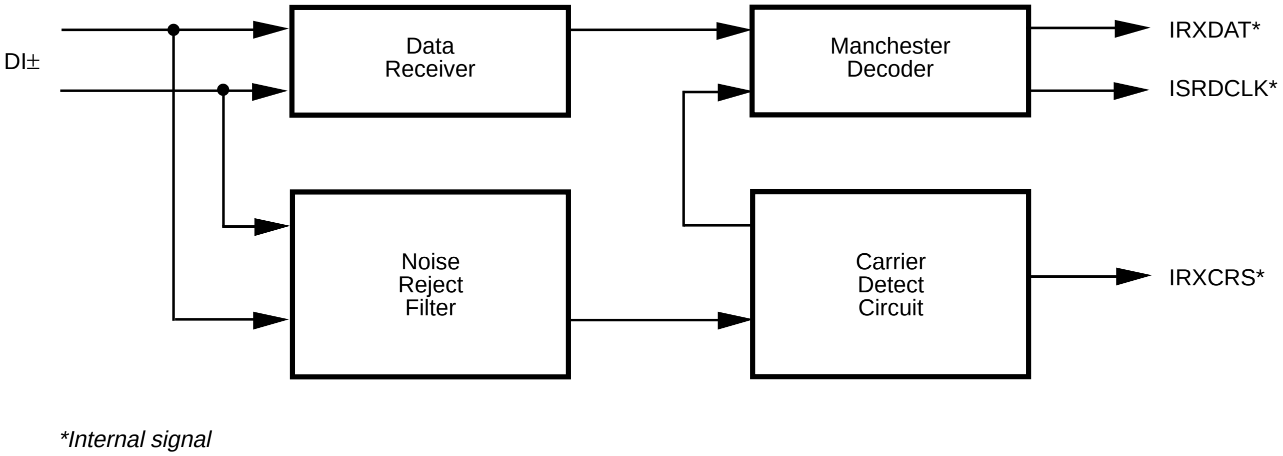
Transmission is enabled by the controller. As long as the ITXEN request remains active, the serial output of the controller will be Manchester encoded and appear at DO±. When the internal request is dropped by the controller, the differential transmit outputs go to one of two idle states, dependent on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

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

### Receiver Path

The principal functions of the Receiver are to signal the Am79C974 controller that there is information on the receive pair, and separate the incoming Manchester encoded data stream into clock and NRZ data.

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



18681A-23

**Figure 19. Receiver Block Diagram**

### Input Signal Conditioning

Transient noise pulses at the input data stream are rejected by the Noise Rejection Filter. Pulse width rejection is proportional to transmit data rate, (which is fixed at 10 MHz for Ethernet but could be different for proprietary networks). DC inputs more negative than minus 100 mV are also surpressed.

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

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

### Clock Acquisition

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

The internal serial receive data clock, ISRDCLK and the internal received data, IRXDAT, are enabled 1/4 bit time after clock acquisition in bit cell 5. IRXDAT is at a HIGH state when the receiver is idle (no ISRDCLK). IRXDAT however, is undefined when clock is acquired and may remain HIGH or change to LOW state whenever ISRDCLK is enabled. At 1/4 bit time through bit cell 5, the controller portion of the Am79C974 controller sees the first ISRDCLK transition. This also strobes in the incoming fifth bit to the MENDEC as Manchester “1”. IRXDAT may make a transition after the ISRDCLK rising edge in bit cell 5, but its state is still undefined. The Manchester “1” at bit 5 is clocked to IRXDAT output at 1/4 bit time in bit cell 6.

### PLL Tracking

After clock acquisition, the phase-locked clock is compared to the incoming transition at the bit cell center (BCC) and the resulting phase error is applied to a correction circuit. This circuit ensures that the phase-locked clock remains locked on the received signal. Individual bit cell phase corrections of the Voltage Controlled Oscillator (VCO) are limited to 10% of the phase difference between BCC and phase-locked clock. Hence, input data jitter is reduced in ISRDCLK by 10 to 1.

### Carrier Tracking and End of Message

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

### Data Decoding

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

### Differential Input Terminations

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

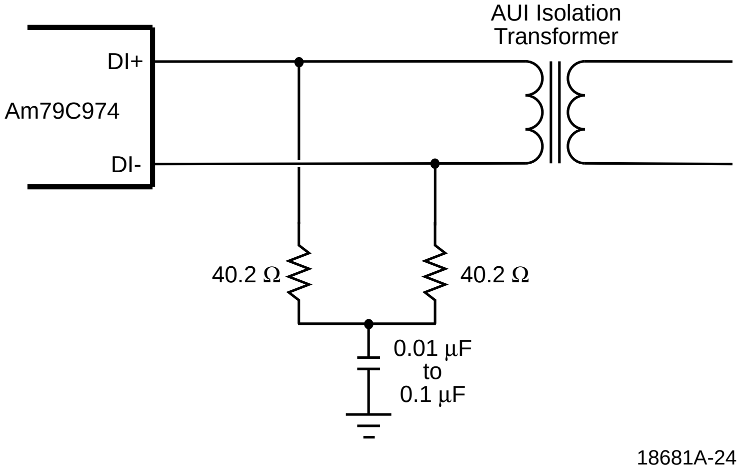


**Figure 20. Differential Input Termination**

### Collision Detection

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

### Jitter Tolerance Definition

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

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

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

### Attachment Unit Interface (AUI)

The AUI is the PLS (Physical Layer Signaling) to PMA (Physical Medium Attachment) interface which effectively connects the DTE to a MAU. The differential interface provided by the Am79C974 controller is fully compliant to Section 7 of ISO 8802-3 (ANSI/IEEE 802.3).

After the Am79C974 controller initiates a transmission it will expect to see data “looped-back” on the DI± pair (when the AUI port is selected). This will internally generate a “carrier sense”, indicating that the integrity of the data path to and from the MAU is intact, and that the MAU is operating correctly. This “carrier sense” signal must be asserted at least 6 bit times before the last transmitted bit on DO± (when using the AUI port). If “carrier sense” does not become active in response to the data transmission, or becomes inactive before the end of transmission, the loss of carrier (LCAR) error bit will be set in the Transmit Descriptor Ring (TMD2, bit 27) after the frame has been transmitted.

## Twisted-Pair Transceiver (T-MAU)

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

### Twisted-Pair Transmit Function

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

### Twisted-Pair Receive Function

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

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

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

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

### Link Test Function

The link test function is implemented as specified by 10BASE-T standard. During periods of transmit pair inactivity, 'Link beat pulses' will be periodically sent over the twisted pair medium to constantly monitor medium integrity.

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

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

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

### Polarity Detection and Reversal

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

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

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

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

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

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

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

### Twisted-Pair Interface Status

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

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

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

### Collision Detect Function

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

### Signal Quality Error (SQE) Test (Heartbeat) Function

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

### Jabber Function

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

### Power Down

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

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

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

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

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

### 10BASE-T Interface Connection

Figure 21 shows the proper 10BASE-T network interface design. Refer to the *Technical Manual* (PID #18738A) for more design details, and refer to Appendix B for a list of compatible 10BASE-T filter/transformer modules.

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

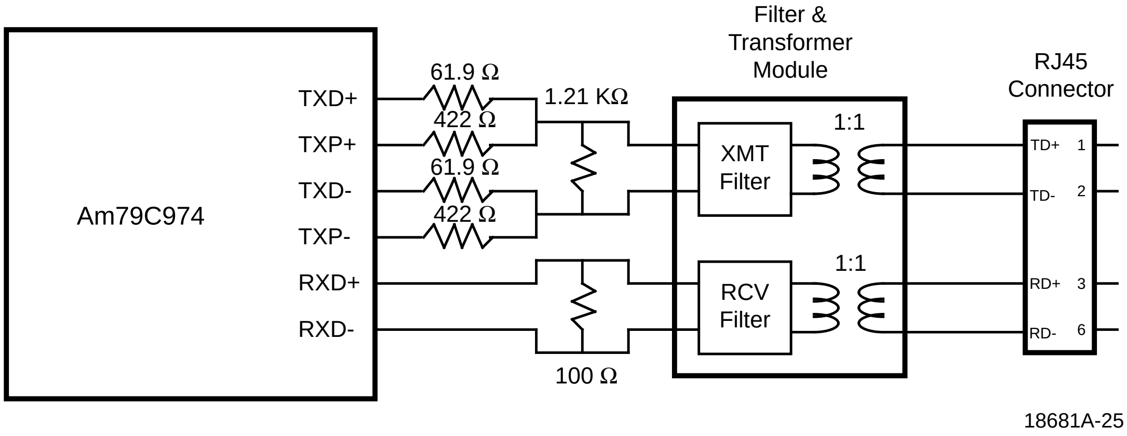


**Figure 21. 10BASE-T Interface Connection**

## Ethernet Power Savings Modes

The Am79C974’s Ethernet controller supports two hardware power savings modes. Both are entered by driving the $\overline{\text{SLEEP}}$ pin LOW.

The PCI interface section is not effected by $\overline{\text{SLEEP}}$. In particular, access to the PCI configuration space remains possible. None of the configuration registers will be reset by $\overline{\text{SLEEP}}$. All I/O accesses to the Am79C974’s Ethernet controller will result in a PCI target abort response.

The first power saving mode is called coma mode. In coma mode, the Am79C974 controller has no means to use the network to automatically wake itself up. Coma mode is enabled when the AWAKE bit in BCR2 is reset. Coma mode is the default power down mode.

The second power saving mode is called snooze mode. In snooze mode, enabled by setting the AWAKE bit in BCR2 and driving the $\overline{\text{SLEEP}}$ pin LOW, the T-MAU receive circuitry will remain enabled even while the $\overline{\text{SLEEP}}$ pin is driven LOW. The $\overline{\text{LNKST}}$ output is the only one of the LED pins that continues to function. The LNKSTE bit must be set in BCR4 to enable indication of a good 10BASE-T link if there are link beat pulses or valid frames present. This $\overline{\text{LNKST}}$ pin can be used to drive an LED and/or external hardware that directly controls the $\overline{\text{SLEEP}}$ pin of the Am79C974 controller. This configuration effectively wakes the system when there is any activity on the 10BASE-T link. Snooze mode can be used only if the T-MAU is the selected network port. Link beat pulses are not transmitted during snooze mode.

If the $\overline{\text{REQ}}$ output is active when the $\overline{\text{SLEEP}}$ pin is asserted, then the Am79C974 controller will wait until the $\overline{\text{GNT}}$ input is asserted. Next, the Am79C974 controller will deassert the $\overline{\text{REQ}}$ pin and finally, it will internally enter either the coma or snooze sleep mode.

Before the sleep mode is invoked, the Am79C974 controller will perform an internal S_RESET. This S_RESET operation will not affect the values of the BCR registers or the PCI configuration space.

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

## Software Access

### Ethernet PCI Configuration Registers

The Am79C974 controller supports the 64-byte header portion of the configuration space as predefined by the PCI specification revision 2.0. None of the device specific registers in locations 64 – 255 are used by the Ethernet controller. The layout of the configuration registers in the header region is shown in the table below. All registers required to identify the Am79C974 controller and its function are implemented. Additional registers are used to setup the configuration of the Am79C974 controller in a system.

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

The configuration registers are accessible only by PCI configuration cycles. They can be accessed right after the Am79C974 controller is powered-on, even if the read operation of the serial EEPROM is still on-going. All multi-byte numeric fields follow little endian byte ordering. The Command register is the only register cleared by H_RESET. S_RESET as well as asserting SLEEP have no effect on the value of the PCI configuration registers. All write accesses to Reserved locations have no effect, reads from these locations will return a data value of ZERO.

When the Am79C974 controller samples its IDSELA or IDSELB input asserted during a configuration cycle, it will acknowledge the cycle by asserting its DEVSEL output. The content of AD[31:00] during the address phase of the configuration cycles must meet the format as shown below:

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

AD[1:0] must both be ZEROs, since the Am79C974 controller is not a bridge device. It only recognizes configuration cycles of Type 0 (as defined by the PCI specification revision 2.0). AD[7:2] specify the selected DWORD in the configuration space. AD[7:6] must both be ZERO, since the Am79C974's Ethernet controller does not implement any of the device specific registers in locations 64 – 255. Since AD[1:0] and AD[7:6] must all be ZERO, the lower 8 bits of the address for a configuration cycle are equal to the offset of the DWORD counting from the beginning of the PCI configuration space. AD[10:8] specify one of eight possible functions of a PCI device. The Am79C974 controller functions as two single function devices, as indicated in the Header Type registers of both PCI configuration spaces (bit 7, FUNCT = 0). Therefore, the Am79C974 controller ignores AD[10:8] during the address phase of a configuration cycle. AD[31:11] are typically used to generate the IDSELA or IDSELB signals. The Am79C974 controller ignores all upper address bits.

PCI configuration registers can be accessed with 8-bit, 16-bit or 32-bit transfers. The active bytes within a DWORD are determined by the byte enable signals. E.G. a read of the Sub-Class register can be performed by reading from offset 08h with only BE2 being active.

### I/O Resources

The Am79C974 controller uses two separate blocks of I/O space, one for the SCSI controller and one for the Ethernet controller. This section discusses the I/O address block used by the Ethernet controller.

### PCnet-SCSI's Ethernet Controller I/O Resource Mapping

The Am79C974's Ethernet controller has several I/O resources. These resources use 32 bytes of I/O space that begin at the Am79C974's Ethernet controller I/O base address.

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

- H_RESET function.
- EEPROM programming of the DWIO mode bit of BCR18.
- Automatic determination of DWIO mode due to DWORD (double-word) I/O write access to offset 10h.

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

Following the H_RESET operation, a PREAD operation of the EEPROM will be executed. If the EEPROM is programmed with a ONE in the DWIO bit position and the EEPROM checksum is correct, then the EEPROM read will result in the setting of the DWIO mode to a ONE.

If the EEPROM programming was absent, failed, or contained a ZERO in the DWIO bit position, then the software may invoke DWIO mode by performing a Double Word write access to the I/O location at offset 10h (RDP). Note that even though the I/O resource mapping changes when the I/O mode setting changes, the RDP location offset is the same for both modes.

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

The mapping of the Ethernet controller resources into the 32-byte I/O space varies depending upon the setting of the DWIO bit of BCR10. Depending upon the setting of this variable, the 32-byte I/O space will be either Word I/O mapped (WIO) or Double Word I/O mapped (DWIO). A DWIO setting of 0 produces Word I/O mode, while a DWIO setting of 1 produces Double Word I/O mapping.

DWIO is automatically programmed as active when the system attempts a DWORD write access to offset 10h of the Ethernet controller I/O space. As long as no DWORD write access to offset 10h of the Ethernet controller I/O space is performed, the Ethernet controller will use the value of DWIO that was programmed from the EEPROM read operation. If no EEPROM is used in the system, then the power up reset value of DWIO will be ZERO, and this value will be maintained until a DWORD access is performed to Ethernet controller I/O space.

Therefore, if DWIO mode is desired, it is imperative that the first access to the Ethernet controller be a DWORD write access to offset 10h, unless the EEPROM will be used to program the DWIO bit.

Alternatively, if DWIO mode is not desired, then it is imperative that the software never executes a DWORD write access to offset 10h of the Ethernet controller I/O space, and the EEPROM programming of the DWIO bit must be ZERO.

Once the DWIO bit has been set to a ONE, only a hardware H_RESET or a new read of the EEPROM can reset it to a ZERO.

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

### *WIO I/O Resource Map*

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

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

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

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

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

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

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

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

### *DWIO I/O Resource Map*

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

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

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

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

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

If an EEPROM is not used to program the value of DWIO, then a DWORD write access to the RDP offset of 10h will automatically program DWIO mode.

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

DWIO mode is exited by asserting the $\overline{\mathrm{RST}}$ pin or by forcing a re-read of the EEPROM when the EEPROM will program a ZERO into the DWIO bit location of BCR10. Assertion of S_RESET or setting the STOP bit of CSR0 will have no effect on the DWIO mode setting.

### *I/O Space Comments*

The following statements apply to both WIO and DWIO mapping:

The RAP is shared by the RDP and the BDP.

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

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

### Am79C974's Ethernet Controller I/O Base Address

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

The contents of the Ethernet I/O Base Address Registers (BCR16 and BCR17) are ignored.

### I/O Register Access

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

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

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

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

### APROM Access

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

When programmed for WIO mode, any byte or word address from an offset of 0h to an offset of Fh may be read. An appropriate byte or word of APROM contents will be delivered by the Am79C974 controller in response to accesses that fall within the APROM range of 0h to Fh.

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

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

Write access to any of the APROM locations is allowed, but only 4 bytes on DWORD boundaries in DWIO mode or 2 bytes on word boundaries in WIO mode (only read accesses to the APROM locations can be in 8-bit quantities while in WIO mode). The IESRWE bit (see BCR2) must be set in order to enable a write operation to the APROM. Only the Am79C974 controller on-board IEEE Shadow registers are modified by writes to APROM locations. The EEPROM is unaffected by writes to APROM locations.

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

### RDP Access (CSR Register Space)

RDP = Register Data Port. The RDP is used with the RAP to gain access to any of the Am79C974 controller CSR locations.

Access to any of the CSR locations of the Am79C974 controller is performed through the Am79C974 controllers Register Data Port (RDP). In order to access a particular CSR location, the Register Address Port (RAP) should first be written with the appropriate CSR address. The RDP now points to the selected CSR. A read of the RDP will yield the selected CSRs data. A write to the RDP will write to the selected CSR.

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

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

### RAP Access

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

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

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

### BDP Access (BCR Register Space)

BDP = Bus Configuration Register Data Port. The BDP is used with the RAP to gain access to any of the Am79C974 controller BCR locations.

Access to any of the BCR locations of the Am79C974 controller is performed through the Am79C974 controllers BCR Data Port (BDP ); in order to access a particular BCR location, the Register Address Port (RAP) should first be written with the appropriate BCR address. The BDP now points to the selected BCR. A read of the BDP will yield the selected BCR s data. A write to the BDP will write to the selected BCR.

When programmed for WIO mode, the BDP has a width of 16 bits, hence, all BCR locations have 16 bits of width in WIO mode. Note that when operating in WIO mode, the upper two bytes of the data bus will be undefined since the byte mask will not be active for those bytes.

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

### RESET Register (S_RESET)

A read of the reset register creates an internal S_RESET pulse in the Am79C974 controller. This read access cycle must be 16 bits wide in WIO mode and 32 bits wide in DWIO mode. The internal S_RESET pulse that is generated by this access is different from both the assertion of the hardware $\overline{\text{RST}}$ pin (H_RESET) and from the assertion of the software STOP bit. Specifically, the reset registers S_RESET will be the equivalent of the assertion of the $\overline{\text{RST}}$ pin (H_RESET) assertion for all CSR locations, but S_RESET will have no effect at all on the BCR or PCI configuration space locations, and S_RESET will not cause a deassertion of the $\overline{\text{REQ}}$ pin.

The NE2100 LANCE based family of Ethernet cards requires that a write access to the reset register follows each read access to the reset register. The Am79C974 controller does not have a similar requirement. The write access is not required but it does not have any harmful effects.

Write accesses to the reset register will have no effect on the Am79C974 controller.

Note that a read access of the reset register will take longer than the normal I/O access time of the Am79C974 controller. This is because an internal S_RESET pulse will be generated due to this access, and the access will not be allowed to complete on the system bus until the internal S_RESET operation has been completed. This is to avoid the problem of allowing a new I/O access to proceed while the S_RESET operation has not yet completed, which would result in erroneous data being returned by (or written into) the Am79C974 controller. The length of a read of the Reset register can be as long as 64 clock cycles.

Note that a read of the reset register will ***not*** cause a deassertion of the $\overline{\text{REQ}}$ signal, if it happens to be active at the time of the read of the reset register. The $\overline{\text{REQ}}$ signal will remain active until the $\overline{\text{GNT}}$ signal is asserted. Following the read of the reset register, on the next clock cycle after the $\overline{\text{GNT}}$ signal is asserted, the Am79C974 controller will deassert the $\overline{\text{REQ}}$ signal. No bus master accesses will have been performed during this brief bus ownership period.

Note that this behavior differs from that which occurs following the assertion of a minimum-width pulse on the $\overline{\text{RST}}$ pin (H_RESET). A $\overline{\text{RST}}$ pin assertion will cause the $\overline{\text{REQ}}$ signal to deassert within six clock cycles following the assertion. In the $\overline{\text{RST}}$ pin case, the Am79C974 controller will not wait for the assertion of the $\overline{\text{GNT}}$ signal before deasserting the $\overline{\text{REQ}}$ signal.

### Vendor Specific Word

This I/O offset is reserved for use by the system designer. The Am79C974 controller will not respond to accesses directed toward this offset. The Vendor Specific Word is only available when the Am79C974 controller is programmed to word I/O mode (DWIO = 0).

If more than one Vendor Specific Word is needed, it is suggested that the VSW location should be divided into a VSW Register Address Pointer (VSWRAP) at one location (e.g. VSWRAP at byte location 18h or word location 30h, depending upon DWIO state) and a VSW Data Port (VSWDP) at the other location (e.g. VSWDP at byte location 19h or word location 32h, depending upon DWIO state). Alternatively, the system may capture RAP data accesses in parallel with the Am79C974 controller and therefore share the Am79C974 controller RAP to allow expanded VSW space. Am79C974 controller will not respond to access to the VSW I/O address.

### Reserved I/O Space

These locations are reserved for future use by AMD. The Am79C974 controller does not respond to accesses directed toward these locations, but future AMD products that are intended to be upward compatible with the Am79C974 controller device may decode accesses to these locations. Therefore, the system designer may not utilize these I/O locations.

## Hardware Access

### PCnet-SCSI Controller Master Accesses

The Am79C974 controller has a bus interface compatible with PCI specification revision 2.0.

Complete descriptions of the signals involved in bus master transactions for each mode may be found in the pin description section of this document. Timing diagrams for master accesses may be found in the block description section for the Bus Interface Unit. This section simply lists the types of master accesses that will be performed by the Am79C974 controller with respect to data size and address information.

The Am79C974 controller will support master accesses only to 32-bit peripherals. The Am79C974 controller does not support master accesses to 8-bit or 16-bit memory. The Am79C974 controller is not compatible with 8-bit systems, since there is no mode that supports Am79C974 controller accesses to 8-bit peripherals.

Table 3 describes all possible bus master accesses that the Am79C974 controller will perform. The right most column lists all operations that may execute the given access:

**Table 3. Bus Master Accesses**

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

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

Note that all Am79C974 controller master read operations will always activate all byte enables. Therefore, no one-, two- or three-byte read operations are indicated in the table.

In the instance where a transmit buffer pointer address begins on a non-DWORD boundary, the pointer will be truncated to the next DWORD boundary address that lies below the given pointer address and the first read access from the transmit buffer will be indicated on the byte enable signals as a four-byte read from this address. Any data from byte lanes that lie outside of the boundary indicated by the buffer pointer will be discarded inside of the Am79C974 controller. Similarly, if the end of a transmit buffer occurs on a non-DWORD boundary, then all byte lanes will be indicated as active by the byte enable signals, and any data from byte lanes that lie outside of the boundary indicated by the buffer pointer will be discarded inside of the Am79C974 controller.

### Slave Access to I/O Resources

The Am79C974 device is always a 32-bit peripheral on the system bus. However, the width of individual software resources on board the Am79C974 controller may be either 16-bits or 32-bits. The Am79C974 controller I/O resource widths are determined by the setting of the DWIO bit as indicated in the following table:

| DWIO Setting | Am79C974 Controller I/O Resource Width | Example Application |
|---|---|---|
| DWIO = 0 | 16-bit | Existing PCnet-ISA driver that assumes 16-bit I/O mapping and 16-bit resource widths |
| DWIO = 1 | 32-bit | New drivers written specifically for the Am79C974 controller |

Note that when I/O resource width is defined as 32 bits (DWIO mode), the upper 16 bits of the I/O resource is reserved and written as ZEROS and read as undefined, except for the APROM locations and CSR88. The APROM locations and CSR88 are the only I/O resources for which all 32 bits will have defined values. However, this is true only when the Am79C974 controller is in DWIO mode.

Configuring the Am79C974 controller for DWIO mode is accomplished whenever there is any attempt to perform a 32-bit write access to the RDP location (offset 10h). See the DWIO section for more details.

Table 4 describes all possible bus slave accesses that may be directed toward the Am79C974 controller. (i.e., the Am79C974 controller is the target device during the transfer.) The first column indicates the type of slave access. RD stands for READ, WR for a WRITE operation. The second column indicates the value of the C/$\overline{\text{BE}}$[3:0] lines during the data phase of the transfer. The four byte columns (AD[31:24], AD[23:16], AD[15:8], AD[7:0]) indicate the value on the address/data bus during the data phase of the access. “data” indicates the position of the active bytes; “copy” indicates the positions of copies of the active bytes; “undef” indicates byte locations that are undefined during the transfer.

**Table 4. Bus Slave Accesses**

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

### EEPROM Microwire Access

The Am79C974 controller contains a built-in capability for reading and writing to an external EEPROM. This built-in capability consists of a Microwire interface for direct connection to a Microwire compatible EEPROM, an automatic EEPROM read feature, and a user-programmable register that allows direct access to the Microwire interface pins.

### Automatic EEPROM Read Operation

Shortly after the deassertion of the RST pin, the Am79C974 controller will read the contents of the EEPROM that is attached to the Microwire interface. Because of this automatic-read capability of the Am79C974 controller, an EEPROM can be used to program many of the features of the Am79C974 controller at power-up, allowing system-dependent configuration information to be stored in the hardware, instead of inside of operating code.

If an EEPROM exists on the Microwire interface, the Am79C974 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 Am79C974 controller. The host can access the PCI Configuration Space during the EEPROM read operations. Access to the Am79C974 I/O resources, however, is not possible during the EEPROM read operation. The Am79C974 controller will terminate these I/O accesses with the assertion of DEVSEL and STOP while TRDY is not asserted, signaling to the initiator to retry the access at a later time.

A checksum verification is performed on the data that is read from the EEPROM. If the checksum verification of the EEPROM data fails, then at the end of the EEPROM read sequence, the Am79C974 controller will force all EEPROM-programmable BCR registers back to their H_RESET default values. The content of the APROM locations (offsets 0h – Fh from the I/O base address), however, will not be cleared. The 8-bit checksum for the entire 36 bytes of the EEPROM should be FFh.

If no EEPROM is present at the time of the automatic read operation, then the Am79C974 controller will recognize this condition and will abort the automatic read operation and reset both the PREAD and PVALID bits in BCR19. All EEPROM-programmable BCR registers will be assigned their default values after H_RESET. The content of the Address PROM locations (offsets 0h – Fh from the I/O base address) will be undefined.

If the user wishes to modify any of the configuration bits that are contained in the EEPROM, then the seven command, data and status bits of BCR19 can be used to write to the EEPROM. After writing to the EEPROM, the host should set the PREAD bit of BCR19. This action forces a Am79C974 controller re-read of the EEPROM so that the new EEPROM contents will be loaded into the EEPROM-programmable registers on board the Am79C974 controller. (The EEPROM-programmable registers may also be reprogrammed directly, but only information that is stored in the EEPROM will be preserved at system power-down.) When the PREAD bit of BCR19 is set, it will cause the Am79C974 controller to terminate further accesses to internal I/O resources with the PCI retry cycle. Accesses to the PCI configuration space is still possible.

#### *EEPROM Auto-Detection*

The Am79C974 controller uses the EESK/LED1 pin to determine if an EEPROM is present in the system. At all rising CLK edges during the assertion of the RST pin, the Am79C974 controller will sample the value of the EESK/LED1 pin. If the sampled value is a ONE, then the Am79C974 controller assumes that an EEPROM is present, and the EEPROM read operation begins shortly after the RST pin is deasserted. If the sampled value of EESK/LED1 is a ZERO, then the Am79C974 controller assumes that an external pulldown device is holding the EESK/LED1 pin low, and therefore, there is no EEPROM in the system. Note that if the designer creates a system that contains an LED circuit on the EESK/LED1 pin but has no EEPROM present, then the EEPROM auto-detection function will incorrectly conclude that an EEPROM is present in the system. However, this will not pose a problem for the Am79C974 controller, since it will recognize the lack of an EEPROM at the end of the read operation, when the checksum verification fails.

#### *Systems Without an EEPROM*

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

- If the LED1 is not needed in the system, then the system designer may connect the EESK/LED1 pin to a resistive pulldown device. This will indicate to the EEPROM auto-detection function that no EEPROM is present.
- If the LED1 function is needed in the system, then the system designer will connect the EESK/LED1 pin to a resistive pullup device and the EEPROM auto-detection function will incorrectly conclude that an EEPROM is present in the system. However, this will not pose a problem for the Am79C974 controller, since it will recognize the lack of an EEPROM at the end of the read operation, when the checksum verification fails.

In either case, following the PCI configuration, additional information, including the ISO 8802-3 (IEEE/ANSI 802.3) station address, may be loaded into the Am79C974 controller. Note that the IESRWE bit (bit 8 of BCR2) must be set before the Am79C974 controller will accept writes to the APROM offsets within the Am79C974 I/O resources map. Startup code in the system BIOS can perform the PCI configuration accesses, the IESRWE bit write, and the APROM writes.

### Direct Access to the Microwire Interface

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

### EEPROM-Programmable Registers

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

| | |
|---|---|
| 1. I/O offsets 0h – Fh | APROM locations |
| 2. BCR2 | Miscellaneous Configuration register |
| 3. BCR16 | not used in the Am79C974 controller |
| 4. BCR17 | not used in the Am79C974 controller |
| 5. BCR18 | Burst Size and Bus Control Register |
| 6. BCR21 | Not Used |

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

Note that accesses to the APROM I/O locations do not directly access the Address EEPROM itself. Instead, these accesses are routed to a set of shadow registers on board the Am79C974 controller that are loaded with a copy of the EEPROM contents during the automatic read operation that immediately follows the H_RESET operation.

### *EEPROM MAP*

The automatic EEPROM read operation will access 18 words (i.e. 36 bytes) of the EEPROM. The format of the EEPROM contents is shown in Table 5, beginning with the byte that resides at the lowest EEPROM address:

**Table 5. EEPROM Contents**

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

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

There are two checksum locations within the EEPROM. The first is required for the EEPROM address. This checksum will be used by AMD driver software to verify that the ISO 8802-3 (IEEE/ANSI 802.3) station address has not been corrupted. The value of bytes 0Ch and 0Dh should match the sum of bytes 00h through 0Bh and 0Eh and 0Fh. The second checksum location – byte 1Fh – is not a checksum total, but is, instead, a checksum adjustment. The value of this byte should be such that the total checksum for the entire 36 bytes of EEPROM data equals the value FFh. The checksum adjust byte is needed by the Am79C974 controller in order to verify that the EEPROM contents have not been corrupted.

## Transmit Operation

The transmit operation and features of the Am79C974 controller are controlled by programmable options. The Am79C974 controller offers a136-byte Transmit FIFO to provide frame buffering for increased system latency, automatic retransmission with no FIFO reload, and automatic transmit padding.

### Transmit Function Programming

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

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

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

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

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

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

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

### Automatic Pad Generation

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

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

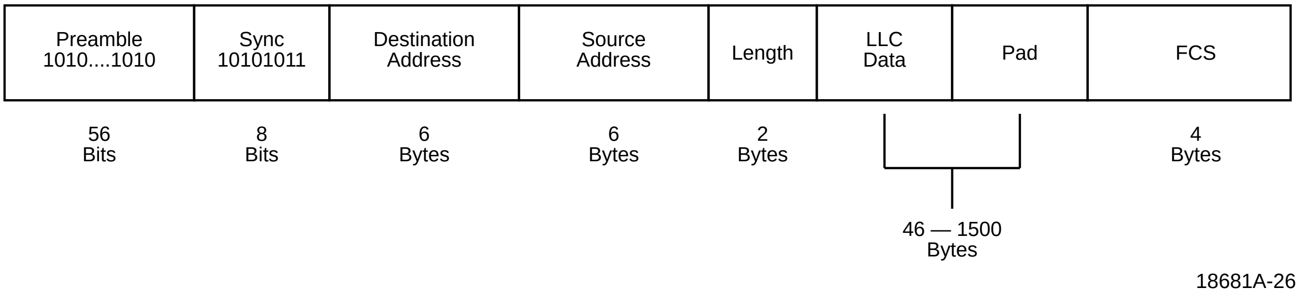


**Figure 22**. **ISO 8802-3 (IEEE/ANSI 802.3) Data Frame**

It is the responsibility of upper layer software to correctly define the actual length field contained in the message to correspond to the total number of LLC Data bytes encapsulated in the packet (length field as defined in the ISO 8802-3 (IEEE/ANSI 802.3) standard). The length value contained in the message is not used by the Am79C974 controller to compute the actual number of pad bytes to be inserted. The Am79C974 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 Am79C974 controller will check to ensure that 544 bits have been transmitted. If not, pad bytes are added to extend the frame size to this value, and the FCS is then added.

The 544 bit count is derived from the following:

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

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

Preamble + (Min Frame Size + FCS) bits

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

Preamble + (Min Frame Size – FCS) bits

64 + (512 – 32) bits

A minimum length transmit frame from the Am79C974 controller will therefore be 576 bits, after the FCS is appended.

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

### Transmit FCS Generation

Automatic generation and transmission of FCS for a transmit frame depends on the value of DXMTFCS bit in CSR15. When DXMTFCS = 0 the transmitter will generate and append the FCS to the transmitted frame. If the automatic padding feature is invoked (APAD_XMT is set in CSR4), the FCS will be appended by the Am79C974 controller regardless of the state of DXMTFCS. Note that the calculated FCS is transmitted most significant bit first. The default value of DXMTFCS is 0 after H_RESET.

### Transmit Exception Conditions

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

Normal events which may occur and which are handled autonomously by the Am79C974 controller include collisions within the slot time with automatic retry. The Am79C974 controller will ensure that collisions which occur within 512 bit times from the start of transmission (including preamble) will be automatically retried with no host intervention. The transmit FIFO ensures this by guaranteeing that data contained within the FIFO will not be overwritten until at least 64 bytes (512 bits) of preamble plus address, length and data fields have been transmitted onto the network without encountering a collision.

If 16 total attempts (initial attempt plus 15 retries) fail, the Am79C974 controller sets the RTRY bit in the current transmit TDTE in host memory (TMD2), gives up ownership (resets the OWN bit to ZERO) for this frame, and processes the next frame in the transmit ring for transmission.

Abnormal network conditions include:

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

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

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

#### *Loss of Carrier*

A loss of carrier condition will be reported if the Am79C974 controller cannot observe receive activity whilst it is transmitting on the AUI port. After the Am79C974 controller initiates a transmission it will expect to see data "looped-back" on the DI± pair. This will internally generate a "carrier sense", indicating that the integrity of the data path to and from the MAU is intact, and that the MAU is operating correctly. This "carrier sense" signal must be asserted about 6 bit times before the last transmitted bit on DO±. If "carrier sense" does not become active in response to the data transmission, or becomes inactive before the end of transmission, the loss of carrier (LCAR) error bit will be set in TMD2 after the frame has been transmitted. The frame will not be re-tried on the basis of an LCAR error.

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

#### *Late Collision*

A late collision will be reported if a collision condition occurs after one slot time (512 bit times) after the transmit process was initiated (first bit of preamble commenced). The Am79C974 controller will abandon the transmit process for the particular frame, set Late Collision (LCOL) in the associated TMD2, and process the next transmit frame in the ring. Frames experiencing a late collision will not be re-tried. Recovery from this condition must be performed by upper layer software.

#### *SQE Test Error*

During the inter packet gap time following the completion of a transmitted message, the AUI CI± pair is asserted by some transceivers as a self-test. The integral Manchester Encoder/Decoder will expect the SQE Test Message (nominal 10 MHz sequence) to be returned via the CI± pair, within a 40 network bit time period after DI± goes inactive (this does not apply if the 10BASE-T port is selected). If the CI± input is not asserted within the 40 network bit time period following the completion of transmission, then the Am79C974 controller will set the CERR bit in CSR0. CERR will be asserted in 10BASE-T mode after transmit if T-MAU is in Link Fail state. CERR will never cause INTA to be activated. It will, however, set the ERR bit CSR0.

## Receive Operation

The receive operation and features of the Am79C974 controller are controlled by programmable options.

### Address Matching

The Am79C974 controller supports three types of address matching: unicast, multicast, and broadcast. The normal address matching procedure can be modified by programming three bits in the MODE register (PROM, DRCVBC, and DRCVPA).

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

When a unicast frame arrives at the Am79C974 controller, the controller will accept the frame if the destination address field of the incoming frame exactly matches the 6-byte station address stored in the PADR registers (CSR12, CSR13, and CSR14). The byte ordering is such that the first byte received from the network (after the SFD) must match the least significant byte of CSR12 (PADR[7:0]), and the sixth byte received must match the most significant byte of CSR14 (PADR[47:40]).

If DRCVPA (bit 13 in the MODE register) is set. the Am79C974 controller will not accept unicast frames.

If the incoming frame is multicast the Am79C974 controller performs a calculation on the contents of the destination address field to determine whether or not to accept the frame. This calculation is explained in the section that describes the Logical Address Filter (LADRF).

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

Although broadcast frames are classified as special multicast frames, they are treated differently by the Am79C974 controller hardware. Broadcast frames are always accepted, except when DRCVBC (bit 14 in the MODE register) is set.

None of the address filtering described above applies when the Am79C974 controller is operating in the promiscuous mode. In the promiscuous mode, all properly formed packets are received, regardless of the contents of their destination address fields. The promiscuous mode overrides the Disable Receive Broadcast bit (DRCVBC bit l4 in the MODE register) and the Disable Receive Physical Address bit (DRCVPA, bit 13 MODE register).

The Am79C974 controller operates in promiscuous mode when PROM (bit 15 in the MODE register) is set.

### Receive Function Programming

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

All receive frames can be accepted by setting the PROM bit in CSR15. When PROM is set, the Am79C974 controller will attempt to receive all messages, subject to minimum frame enforcement. Promiscuous mode over rides the effect of the Disable Receive Broadcast bit on receiving broadcast frames.

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

### Automatic Pad Stripping

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

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

Since any valid Ethernet Type field value will always be greater than a normal 802.3 Length field (≥46), the Am79C974 controller will not attempt to strip valid Ethernet frames.

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

Figure 23 shows the byte/bit ordering of the received length field for an 802.3 compatible frame format.

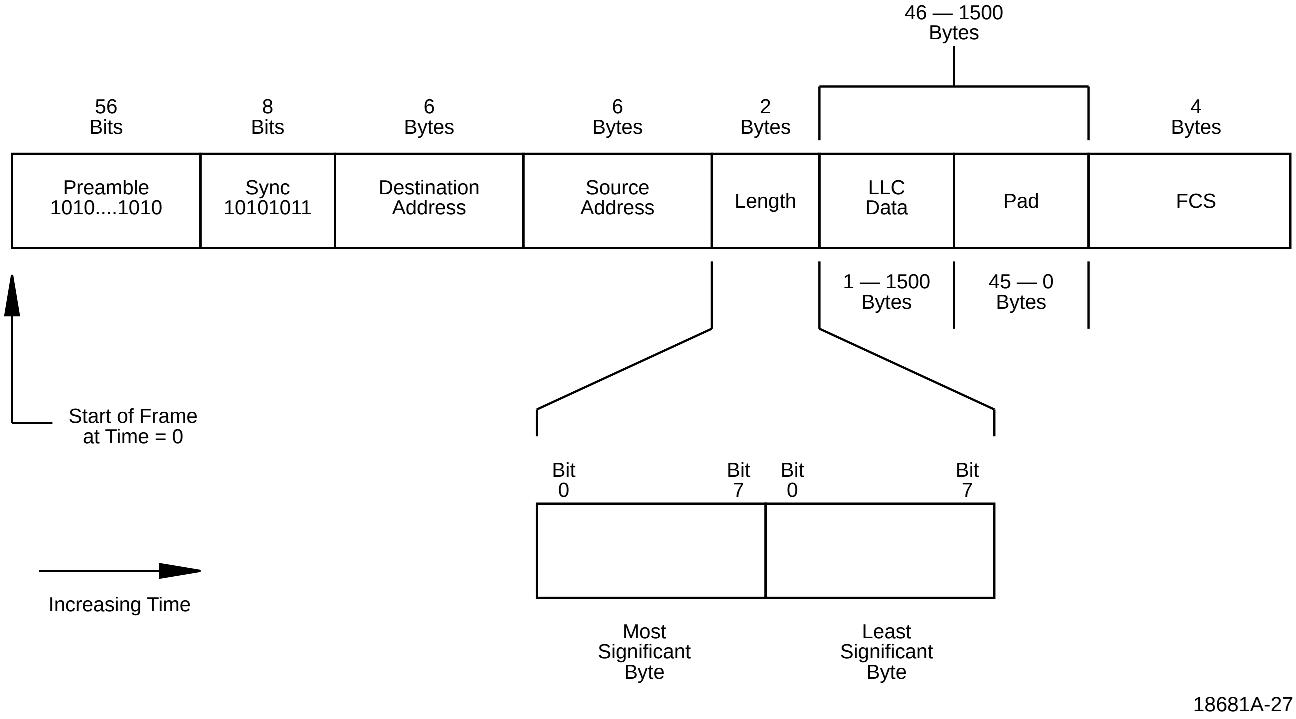


**Figure 23. 802.3 Frame and Length Field Transmission Order**

### Receive FCS Checking

Reception and checking of the received FCS is performed automatically by the Am79C974 controller. Note that if the Automatic Pad Stripping feature is enabled, the FCS for padded frames will be verified against the value computed for the incoming bit stream including pad characters, but the FCS value for a padded frame will not be passed to the host. If an FCS error is detected in any frame, the error will be reported in the CRC bit in RMD1.

### Receive Exception Conditions

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

Normal events which may occur and which are handled autonomously by the Am79C974 controller are basically collisions within the slot time and automatic runt packet rejection. The Am79C974 controller will ensure that collisions which occur within 512 bit times from the start of reception (excluding preamble) will be automatically deleted from the receive FIFO with no host intervention. The receive FIFO will delete any frame which is composed of fewer than 64 bytes provided that the Runt Packet Accept (RPA bit in CSR124) feature has not been enabled. This criterion will be met regardless of whether the receive frame was the first (or only) frame in the FIFO or if the receive frame was queued behind a previously received message.

Abnormal network conditions include:

- FCS errors
- Late Collision

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

## Loopback Operation

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

There are restrictions on loopback operation. The Am79C974 controller has only one FCS generator circuit. The FCS generator can be used by the transmitter to generate the FCS to append to the frame, or it can be used by the receiver to verify the FCS of the received frame. It can not be used by the receiver and transmitter simultaneously.

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

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

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

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

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

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

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

When performing an internal loopback, no frame will be transmitted to the network. However, when the Am79C974 controller is configured for internal loopback the receiver will not be able to detect network traffic. External loopback tests will transmit frames onto the network if the AUI port is selected, and the Am79C974 controller will receive network traffic while configured for external loopback when the AUI port is selected. Runt Packet Accept is automatically enabled when any loopback mode is invoked.

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

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

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

## LED Support

The Am79C974 controller can support up to 3 LEDs.

LED outputs $\overline{\text{LNKST}}$ and $\overline{\text{LED1}}$ allow for direct connection of an LED and its supporting pullup device. LED output $\overline{\text{LED3}}$ may require an additional buffer between the Am79C974 controller output pin and the LED and its supporting pullup device.

Because the $\overline{\text{LED3}}$ output is multiplexed with other Am79C974 controller functions, it may not always be possible to connect an LED circuit directly to the $\overline{\text{LED3}}$ pin. For example, when an LED circuit is directly connected to the EEDO/$\overline{\text{LED3}}$ pin, then it is not possible for most serial EEPROM devices to sink enough $I_{OL}$ to maintain a valid low level on the EEDO input to the Am79C974 controller. Therefore, in applications that require both an EEPROM and a third LED, then it is necessary to buffer the $\overline{\text{LED3}}$ circuit from the EEPROM-PCnet-SCSI connection. The LED registers in the BCR resource space allow each LED output to be programmed for either active high or active low operation, so that both inverting and non-inverting buffering choices are possible.

In applications where an EEPROM is not needed, the $\overline{\text{LED3}}$ pin may be directly connected to an LED circuit. The Am79C974 $\overline{\text{LED3}}$ pin driver will be able to sink enough current to properly drive the LED circuit.

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

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

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

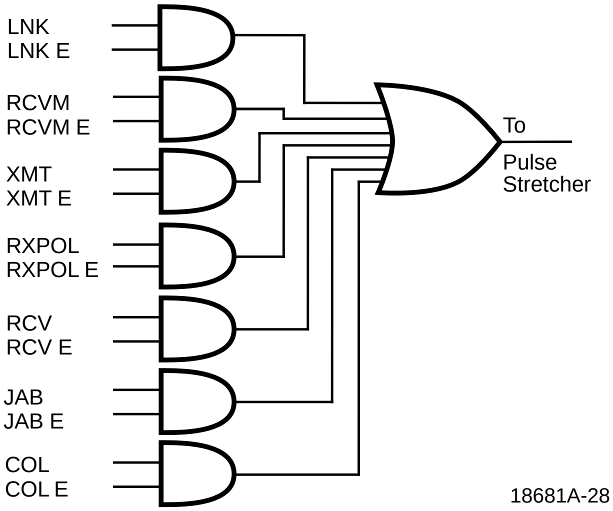


**Figure 24. LED Control Logic**

The diagram above shows the LED signal circuit that exists for each LED pin within the Am79C974 controller.

## H_RESET, S_RESET, and STOP

There are three different types of RESET operations that may be performed on the Am79C974 device, H_RESET, S_RESET and STOP. These names have been used throughout the document. The following is a description of each type of RESET operation:

### H_RESET

H_RESET= HARDWARE_RESET is a Am79C974 RESET operation that has been created by the proper assertion of the $\overline{\text{RST}}$ PIN of the Am79C974 device. When the minimum pulse width timing as specified in the $\overline{\text{RST}}$ pin description has been satisfied, then an internal RESET operation will be performed.

H_RESET will RESET all of or some portions of CSR0, 3, 4, 15, 58, 80, 82, 100, 112, 114, 122, 124 and 126 to default values. H_RESET will RESET all of or some portions of BCR 2, 4, 5, 6, 7, 18, 19, 20, 21 to default values. H_RESET will reset the Command register in the PCI configuration space. H_RESET will cause the microcode program to jump to its RESET state. Following the end of the H_RESET operation, the Am79C974 controller will attempt to read the EEPROM device through the EEPROM Microwire interface. H_RESET resets the T-MAU into the link fail state.

### S_RESET

S_RESET = SOFTWARE_RESET is an Ethernet controller RESET operation that has been created by a read access to the RESET REGISTER which is located at offset 14hex from the Am79C974 I/O base address.

S_RESET will RESET all of or some portions of CSR0, 3, 4, 15, 80, 100 and 124 to default values. S_RESET will not affect any of the BCR and PCI configuration space locations. S_RESET will cause the microcode program to jump to its RESET state. Following the end of the S_RESET operation, the Am79C974 controller will NOT attempt to read the EEPROM device. S_RESET sets the T-MAU into the link fail state.

Note that S_RESET will not cause a deassertion of the $\overline{\text{REQ}}$ signal, if it happens to be active at the time of the read to the reset register. The $\overline{\text{REQ}}$ signal will remain active until the $\overline{\text{GNT}}$ signal is asserted. Following the read of the RESET register, on the next clock cycle after the $\overline{\text{GNT}}$ signal is asserted, the Am79C974 controller will deassert the $\overline{\text{REQ}}$ signal. No bus master accesses will have been performed during this brief bus ownership period.

### STOP

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

STOP will RESET all or some portions of CSR0, 3, and 4 to default values. STOP will not affect any of the BCR and PCI configuration space locations. STOP will cause the microcode program to jump to its RESET state. Following the end of the STOP operation, the Am79C974 controller will NOT attempt to read the EEPROM device. For the identity of individual CSRs and bit locations that are affected by STOP, see the individual CSR register descriptions. Setting the STOP bit does not affect the T-MAU.

## SCSI Controller

The primary function of the PCnet-SCSI controller is to transfer data between the 4 byte-wide PCI bus and 1 byte-wide SCSI bus. The controller consists of two blocks: SCSI and DMA. The SCSI block sits between the SCSI bus and the DMA block. It controls data flow to/from SCSI bus. The DMA block is located between the SCSI block and the PCI bus Interface Unit. It handles data flow to/from PCI bus.

The operation of each block is governed by a set of control registers:

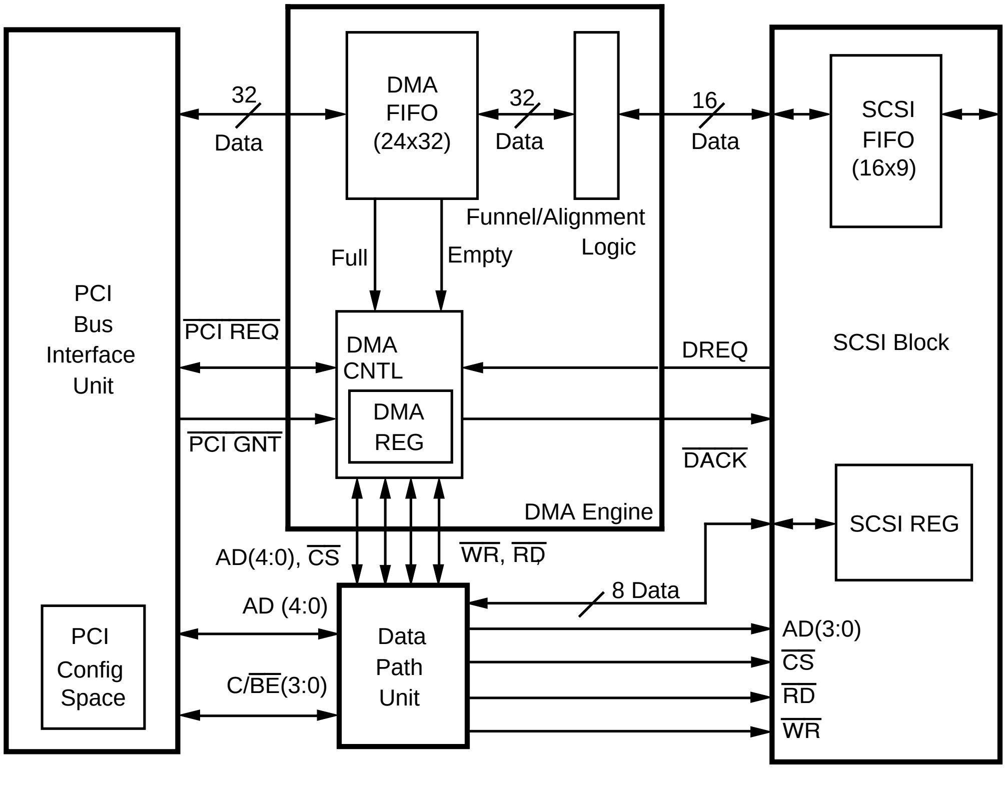
1. Channel Context Block (CCB) registers control the DMA block
2. SCSI registers control the SCSI block

In a normal operation, both sets of registers must be programmed with the specifics of the transfer, such as starting address, transfer count, etc. (For more information, refer to *Technical Manual* PID #18738A).

## SCSI Specific DMA Engine

The SCSI Specific DMA Engine in the Am79C974 provides bus-mastering capabilities to allow flexibility and performance advantages over slave PCI-SCSI devices. Built into the engine is a 96-byte (24 DWORD) FIFO and additional logic to handle the transition between the 32-bit PCI bus and the 8-bit SCSI bus.

Figure 25 illustrates the DMA Engine in relation to the PCI interface and the SCSI block. As its most basic function, the DMA engine acts as the DMA controller in a bus master capacity on the PCI bus, transferring data between memory and the SCSI block. All Command, Data, Status, and Message bytes pass through the DMA FIFO on their way to or from the SCSI bus. However, for programmed I/O (PIO) accesses to the SCSI registers, the DMA FIFO is bypassed as data moves directly from the SCSI block to the PCI interface. Since PIO operations do not pass through the funneling logic and DMA FIFO, data is transferred one byte at a time from the SCSI block to the PCI interface via the least significant byte lane. (The three most significant byte lanes will contain null data.)



18681A-29

**Figure 25. PCI BIU – DMA Engine – SCSI Block**

Since the PCI bus is 4 bytes wide and the SCSI bus is only 1 byte wide, funneling logic is included in this engine to handle byte alignment and to ensure that data is properly transferred between the SCSI bus and the wider PCI bus. All boundary conditions are handled through hardware by the DMA Engine.

The DMA engine is also designed for block type (4 KByte page) transfers to support scatter-gather operations. Implementation of this feature is described further in the DMA Scatter-Gather Mechanism section.

### DMA FIFO

Data transfers from the SCSI FIFO to the DMA FIFO take place each time the threshold of two bytes is reached on the SCSI side. The transfer is initiated by the SCSI block when the internal DREQ is asserted, and continues with the $\overline{\text{DACK}}$ handshaking which typically takes place in DMA accesses. Data is accumulated in the DMA FIFO until a threshold of 16 DWORD (64 bytes) is reached. Data is then burst across the PCI bus to memory. Residue data which is less than the threshold in each FIFO is sent in non-contiguous bursts. For memory read operations, data is sent in burst mode to the DMA FIFO and continues through to the SCSI FIFO and onto the SCSI bus.

### DMA BLAST Command

This command is used to retrieve the contents of the DMA FIFO when the Target disconnects during a DMA Write operation. This could happen for example if a SCSI disk drive detected the end of a sector and decided to give up the bus while it was looking for the next sector. The Target Disconnect can leave some bytes of data in the DMA FIFO and some in the SCSI FIFO, while some bytes have yet to be transferred from the peripheral device. When this happens, the controller will assert $\overline{\text{INTA}}$ to interrupt the processor, the SCSI state machine will continue to empty its contents into the DMA FIFO, but the DMA FIFO will not necessarily dump its contents into memory (unless the 64-byte DMA threshold happens to have been exceeded at this time).

The BLAST command causes the contents of the DMA FIFO to be emptied into memory. There are some restrictions on when this command should be used.

- First, the command should be used only to recover from an interrupted DMA write operation—not a read operation.
- Second, the command must not be issued until the SCSI FIFO has finished dumping its contents into the DMA FIFO.
- Third, the command should never be issued when the DMA FIFO has already been emptied. This is indicated by the state of the DONE bit in the DMA STATUS register at ((B)+54h).

(This is a test for a special case that can occur when a Target Disconnect leaves only 1 byte left to be transferred from the SCSI peripheral. In this case, if the original transfer count was even, an odd number of bytes will be left in the SCSI FIFO. Since the SCSI engine transfers data to the DMA FIFO two bytes at a time, the last transfer consists of one byte of valid data and one byte of garbage. The DMA engine treats this final invalid byte as valid data and writes it to memory. When it does this, it decrements its Working Byte Counter to zero and sets the DONE bit, even though 1 byte still needs to be retrieved from the peripheral device.)

The following procedure outlines the use of the BLAST command after an interrupt has occurred. Note that the order of steps 2–4 is not critical. The order can be changed to tune the performance. Also note that each register is read only once in this procedure even though several tests may be made on data from one register.

1. Verify that INT (bit 7 of the SCSI status register at ((B)+10h) is set to indicate that a SCSI interrupt is pending.
2. Read the SCSI current FIFO count (bits 4:0 of the Current FIFO/Internal State register at ((B)+1Ch). If this value is not zero, wait for the SCSI FIFO to empty its contents into the DMA FIFO.
3. If bit 4 of the SCSI status register (CTZ) is set, stop here. The transfer is complete, and it is not necessary to execute the BLAST command.
4. Verify that DIR (bit 7 of DMA command register at ((B)+40h) is set to one to indicate that the direction of transfer is from SCSI to memory.
5. Test the error bits in the DMA status register and the SCSI status register (STATREG at (B)+10h) to verify that the contents of the DMA and SCSI FIFOs are not invalid.
6. Test the DMA DONE bit in the DMA STATUS register. If DONE is not set, write ‘01’ to the DMA command register to issue the BLAST command. This will move the remaining data from the DMA FIFO into memory.
7. Wait until the BLAST complete (BCMPLT) in the DMA STATUS register is set to indicate the completion of the BLAST operation.
8. Write ‘00’ to the DMA command register to issue the IDLE command to the DMA engine. (Note that the IDLE command does not generate an interrupt.)

The above procedure insures that the data that has been transferred from the SCSI peripheral does not get lost in the DMA FIFO when a Target Disconnect occurs. However, it does not complete the original transfer. The software must now read the SCSI Current Transfer Count register (CTCREG) to find out how many bytes have yet to be transferred from the SCSI peripheral device and must start a new transfer operation to get the rest of the data. (CTCREG consists of three bytes located at ((B)+00h, (B)+04h, and (B)+38h.)

### Funneling Logic

Figure 26 shows the internal DMA logic interface with the SCSI block. The DMA FIFO interfaces to the Funnel

Logic block via a 32-bit data bus, and the funnel logic properly reduces this stream of data to a 16-bit stream to properly interface with the SCSI FIFO.

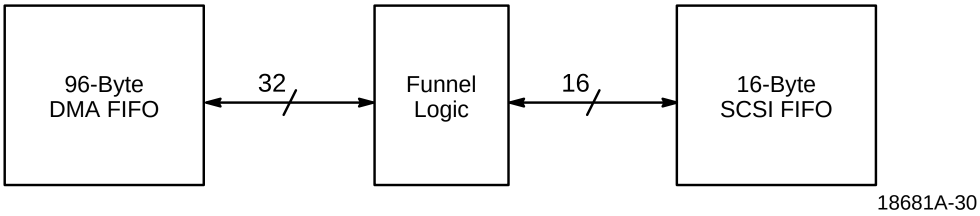


**Figure 26. DMA FIFO to SCSI FIFO Interface**

### SCSI DMA Programming Sequence

The following section outlines the procedure for executing SCSI DMA operations:

1. Issue IDLE command to the DMA Engine
2. Configure the SCSI block registers (e.g. synchronous operation, offset values, etc.)
3. Program the DMA registers to set up address and transfer count
4. Issue a transfer command to the SCSI command registers
5. Issue the START command to the DMA engine
6. At the end of the DMA transaction, issue the IDLE command to the DMA engine

### Memory Descriptor List (MDL) Based DMA Programming

The following section outlines the use of the MDL for scatter-gather DMA operations:

1. Set up the MDL list
2. Use the programming sequence defined earlier for initiating a SCSI DMA transfer

## DMA Registers

The following is a summary of the DMA register set or the DMA Channel Context Block (DMA CCB). These registers control the specifics for DMA operations such as transfer length and scatter-gather options. The three read-only working counter registers allow the system software and driver to monitor the DMA transaction. Each register address is represented by the PCI Configuration Base Address (B) and its corresponding offset value. The Base address for the SCSI controller is stored at register address (10h) in the SCSI PCI configuration space.

**Table 6. The DMA Registers**

| Register Acronym | Addr (Hex) | Register Description | Type |
|---|---|---|---|
| CMD | (B)+40 | Command (bits 31:8 reserved, bits 7:0 used) | R/W |
| STC | (B)+44 | Starting Transfer Count (bits 31:24 reserved, bits 23:0 used) | R/W |
| SPA | (B)+48 | Starting Physical Address (bits 31:0 used) | R/W |
| WBC | (B)+4C | Working Byte Counter | R |
| WAC | (B)+50 | Working Address Counter (bits 31:0 used) | R |
| STATUS | (B)+54 | Status Register (bits 31:8 reserved, bits 7:0 used) | R |
| SMDLA | (B)+58 | Starting Memory Descriptor List (MDL) Address | R/W |
| WMAC | (B)+5C | Working MDL Counter | R |

## Command Register (CMD)

The upper 3 bytes of Command register are reserved, the remaining (LSB) byte is defined as follows:

**Address (B)+40h, LSB** **READ/WRITE**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DIR | INTE_D | INTE_P | MDL | Reserved | Reserved | CMD1 | CMD0 |

### *DIR:*

Data transfer direction bit. When this bit is set, the direction of transfer is from SCSI to memory.

### *INTE_D:*

DMA transfer active interrupt enable bit. When this bit is set, the ERROR or DONE condition will cause $\overline{\text{INTA}}$ to be asserted.

### *INTE_P:*

Page transfer active interrupt bit.

### *MDL:*

Memory Descriptor List (MDL) SPA enable bit.

### *RESERVED:*

Reserved for future expansion. The zero value must be written in these bits.

### *CMD1-0:*

These two bits are encoded to represent four commands: IDLE, BLAST, START, and ABORT.

| CMD1 | CMD0 | Command | Description |
|---|---|---|---|
| 0 | 0 | IDLE | Resets the DMA block to the IDLE state. Stops any current transfer. Does not affect status bits or cause an interrupt. |
| 0 | 1 | BLAST | Empties all data bytes in DMA FIFO to memory during a DMA write operation. Upon completion, the 'BCMPLT' bit will be set in the DMA Status register. This command should not be used during a DMA read operation. |
| 1 | 0 | ABORT | Terminates the current DMA transfer. Restores the DMA engine to the IDLE state. Sets the ABORT bit (bit 2) in the status register.<br>*Note: This is only valid after a 'START' command is issued.* |
| 1 | 1 | START | Initiates a new DMA transfer. These bits must remain set throughout the DMA operation until the 'DONE' bit in the DMA Status Register is set.<br>*Note: This command should be issued only after all other control bits have been initialized.* |

### DMA Scatter-Gather Mechanism

The Am79C974 controller contains a scatter-gather translation mechanism which facilitates faster data transfers. This feature uses a **M**emory **D**escriptor **L**ist which is stored in system memory. Use of the Memory Descriptor List allows a single SCSI transfer to be read from (or written to) non-contiguous physical memory locations. This mechanism avoids copying the transfer data and MDL list, which was previously required for conventional DMA operations.

### *Memory Descriptor List (MDL)*

The MDL is a non-terminated (no End Of File marker) list of 32-bit page frame addresses, which is always aligned on a Double Word boundary. The format is shown below:

| 31 12 | 11 0 |
|---|---|
| Page Frame Address | Ignored |

### *DMA Scatter-Gather Operation (4k aligned elements)*

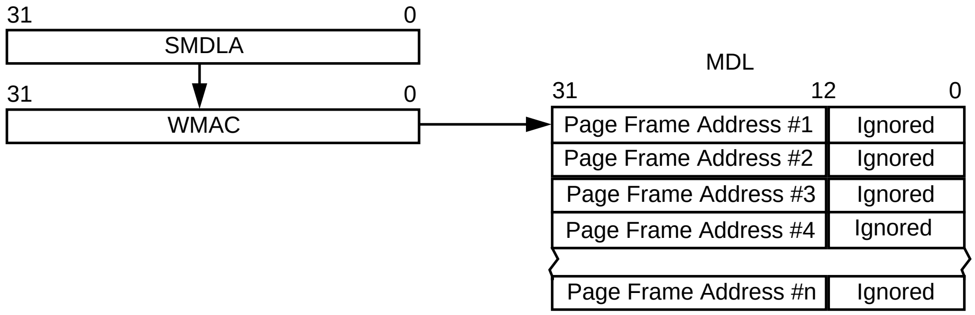
The scatter-gather mechanism described below assumes 4k page alignment and size for all MDL entries except the first and last entry. This feature is enabled by setting the MDL bit in the DMA Command register (Bit 4, Address (B)+40h).

1. a) Prepare the Memory Descriptor List (MDL) through software and store it in system memory.

   b) Load the address of the starting entry in the Memory Descriptor List (MDL) into the Start Memory Descriptor List Address (SMDLA) register. This value is automatically copied into the Working MDL Address Counter (WMAC).

   c) Program the Starting Transfer Count (STC) register with the total transfer length (i.e., # of bytes). Also program the Starting Physical Address (SPA) register (bits 11:0) with the starting offset of the first entry.

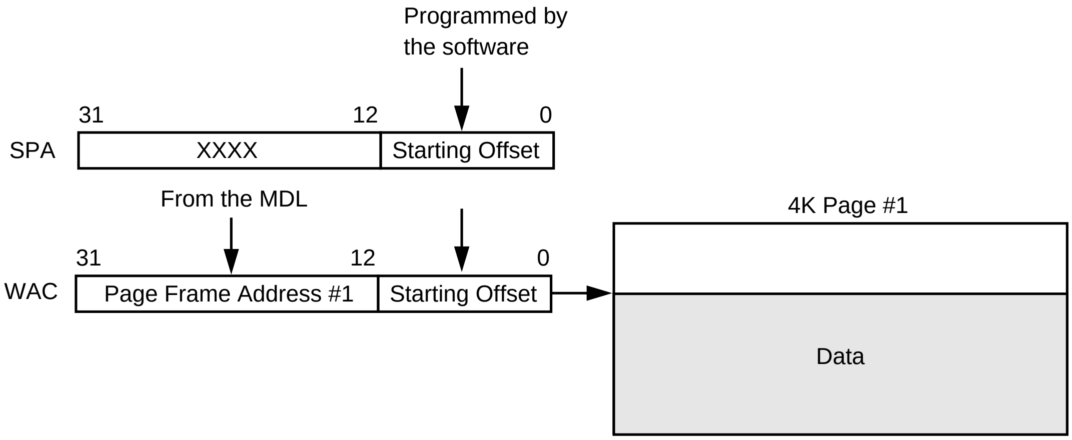
***Note**: The value in the SMDLA register must be double word aligned. Therefore, read/write transactions will always begin on a double word boundary.*



18681A/1-31

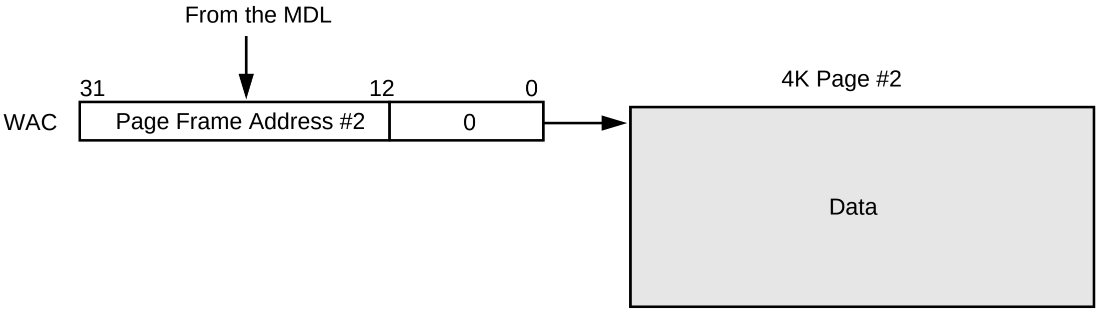
In this example, the contents of the WMAC register is pointing to page frame address #1. When the first entry in the MDL is read (page frame address #1), the WMAC register is incremented to point to the next page entry (page frame address #2).

2. Issue the Start DMA Command. The SCSI controller reads the page frame address (bits 31:12) from the MDL entry and combines it with the first page offset value in the Starting Physical Address (SPA) register (bits 11:0). This 32-bit value is loaded into the Working Address Counter (WAC) register and becomes the physical address for page#1, as shown below. The WMAC is then incremented to point to the next entry in the MDL.



18681A/1-32

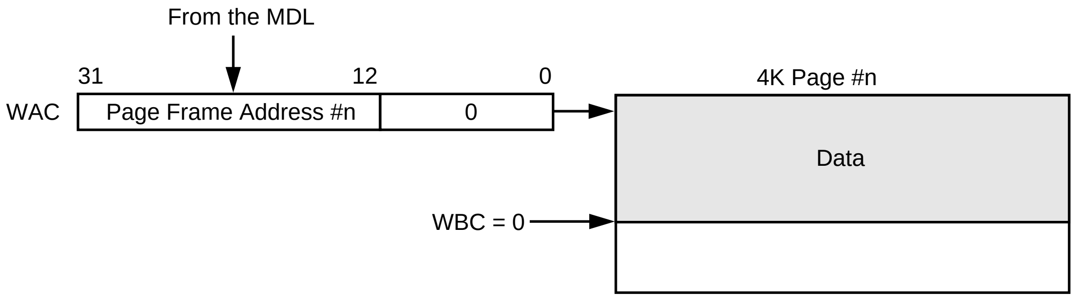
3. When the WAC register (bits 11:0) reaches the first 4K byte boundary, the SCSI controller reads the second MDL entry and combines the page frame address (bits 31:12) from this entry of the MDL with bits 11:00 of the WAC register. This becomes the physical address for page#2. Since the WAC register (bits 11:0) has rolled over to ‘00h’, the WAC now points to the beginning of the Page Frame Address #2 as shown below. The WMAC is then incremented to point to the next entry in the MDL.



18681A/1-33

When WAC (bits 11:0) again reaches the next 4K byte boundary, the next MDL entry is read into the WAC. The operation continues in this way until WMAC register reaches the last MDL entry (Page Frame Address #n in this example).

4. The WAC register points to the beginning of the last page and the DMA operation continues until the byte count is exhausted in the Working Byte Counter (WBC) register. When WBC=0, the chip stops incrementing the WAC register. This is shown below.



18681A/1-34

***DMA Scatter-Gather Operation (Non-4k aligned elements MDL not set)***

There is another way to implement a scatter-gather operation which does not force the data elements to be aligned on 4k boundaries. It assumes a “traditional” scatter-gather list of the following format:

| | |
|---|---|
| Element 0 | Physical Address Byte Count |
| Element 1 | Physical Address Byte Count |
| | ... |
| Element n | Physical Address Byte Count |

This second implementation is described as follows:

1. Set the SCSI Start Transfer Count Register ((B)+00h, (B)+04h, (B)+38h) to the Byte count of the first Scatter-Gather element.
2. Program the DMA Starting Transfer Count Register ((B)+44h) to the Byte Count of the first Scatter-Gather element.
3. Program the DMA Starting Physical Address Register ((B)+48h) to the Physical Address of the first Scatter-Gather element.
4. Start the SCSI operation by issuing a SCSI Information Transfer command.
5. Start the DMA Engine with DMA Transfer Interrupt Enable (Bit 6, (B)+40h).
6. When the Scatter-Gather element’s Byte Count is exhausted, the DMA engine will generate an interrupt.
7. Reprogram the next Scatter-Gather element’s Byte Count into the SCSI Start Transfer Count Register and the DMA Starting Transfer Count Register.
8. Reprogram the DMA Starting Physical Address Register ((B)+48h) to the Physical Address of the next Scatter-Gather element.
9. Repeat steps 4–8 until the Scatter-Gather list is completed.

## Interrupts

Interrupts may come from two sources: the DMA engine or the SCSI block. Upon receipt of an interrupt ($\overline{\text{INTA}}$ asserted), the DMA Status register should be serviced first to identify the interrupt source(s). DMA engine related interrupts are cleared when the related flags are read in the DMA Status register. However, SCSI block interrupts will be cleared only when the SCSI Status, Internal State, and Interrupt Status Registers are read.

Interrupts are caused by:

- Successful completion of a DMA transfer request. (Bit 6 in the DMA Command Register ((B+40h) must be set to enable this interrupt)
- An address error occurred on the PCI bus during a DMA transfer (Bit 6 in the DMA Command Register ((B)+40h) must be set to enable this interrupt)
- The PWDN pin is first asserted
- After completion of each page transfer during MDL operations. (Bit 5 in the DMA Command Register ((B)+40h) must be set to enable this interrupt)

An interrupt from the SCSI block will automatically set bit 4 (SCSIINT) in the DMA Status register (Address (B)+54h). The SCSI block will generate an interrupt under the following conditions:

- SCSI Reset occurred

- Illegal command code issued
- The target disconnects from the SCSI bus
- SCSI bus service request
- Successful completion of a command
- The Am79C974 has been reselected

## The Fast SCSI Block

The functionality of the SCSI block is described in the following section. Topics to be covered are:

- SCSI Block ID
- SCSI FIFO Threshold
- Data Transmission
- $\overline{\text{REQ}}$/$\overline{\text{ACK}}$ Control
- Parity
- Reset Levels

### SCSI Block ID

The Am79C974 contains a SCSI Block ID code which is stored in the MSB of the Current Transfer Count Register. The code reflects the chip's revision level and family code. This 8-bit code may be read when the following conditions are true.

- After power up or a chip reset has occurred
- Before the Current Transfer Counter ((B)+38h) is loaded

The part-unique ID code in Register ((B)+38h) will read as follows:

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

### SCSI FIFO Threshold

The threshold value for the SCSI FIFO is two bytes (one word). When this threshold is reached, the SCSI block will indicate to the DMA engine that it is capable of receiving or sending data bytes.

### Data Transmission

Data transmission rates will vary from system to system, depending on the number of devices configured on the SCSI bus, as well as the transfer rates that each individual device is capable of sustaining.

Transfer rates for the Am79C974 are controlled by the FASTSCSI and FASTCLK bits in Control Register Three, as well by the Extended Timing Feature in Control Register One.

To achieve 10 Mbyte/s transmission rates, the following conditions must be true:

- A 40 MHz clock (50% duty cycle) must be supplied to SCSICLK.
- The target must be able to sustain Fast SCSI timings.
- Bits 3 (FASTCLK) and 4 (FASTSCSI) in Control Register Three must be set to '1.'
- The lower three bits of Register ((B)+24h), the Clock Conversion Factor Register must be programmed to '000.'
- The lower 5 bits of the Synchronous Transfer Period Register ((B)+18h) must be set to a value of '04h.'

The FASTCLK and FASTSCSI bits in Control Register Three modify the SCSI state machine to produce both FAST and Normal SCSI timings. Synchronous data transmission rates are dependent on the input clock frequency selected, as well as the transfer period.

Bits 4:0 in the Synchronous Transfer Period Register ((B)+18h) specify the period for synchronous data transfers. For programming information, refer to the Technical Manual.

### $\overline{\text{REQ}}$/$\overline{\text{ACK}}$ Control

The assertion and deassertion time for the $\overline{\text{REQ}}$ and $\overline{\text{ACK}}$ signals may be controlled via the Synchronous Offset Register ((B)+1Ch). Bits 7:6 control $\overline{\text{REQ}}$/$\overline{\text{ACK}}$ deassertion delay with respect to the time that data is valid, while bits 5:4 control $\overline{\text{REQ}}$/$\overline{\text{ACK}}$ assertion delay. The deassertion for $\overline{\text{REQ}}$/$\overline{\text{ACK}}$ may be moved ahead .5 clock cycles, or it may be delayed for up to 1.5 clock cycles. Deassertion delay options depend on the status of the FASTCLK bit in Control Register Three. Assertion delay for $\overline{\text{REQ}}$/$\overline{\text{ACK}}$ can vary from 0 to 1.5 clock cycles.

### Parity

Parity on the SCSI bus is such that the total number of logical ones on data bus including the parity bit must be odd. Parity checking features are implemented via two bits in the Status Register and Control Register One. Parity checking can be implemented on data flowing in from the SCSI bus. Parity is always generated internally by the Am79C974 for data moving onto the SCSI bus.

| Feature | Bit Name | Bit # | Register |
|---|---|---|---|
| Parity From SCSI | Parity Error Reporting | 4 | Control Reg One ((B)+20h) |
| Parity Status | Parity Error | 5 | Status Register ((B)+10h) |

### Parity Checking on the SCSI Bus

The Parity Error Reporting Bit (Bit 4, Control Register One) enables parity checking on all incoming bytes from the SCSI bus. This feature is cleared to '0' by a hardware reset.

When this feature is enabled, the Am79C974 will check parity on all data received from the SCSI bus. Any detected error will be flagged by setting bit 5 in the SCSI Status Register, and ATN will be asserted on the SCSI bus. However, no interrupt will be generated.

When this feature is disabled (bit 4 set to ‘0’), no parity check is done on incoming bytes. Note that the parity on the PCI bus is generated internally and is distinct from the parity received from the SCSI bus.

### Parity Generating on the SCSI Bus

For each byte transferred to the SCSI bus, parity generation is done automatically.

## Reset Levels

The Am79C794 has two reset pins and two reset commands that affect the SCSI block. The RST pin resets the whole chip including the SCSI controller, the Ethernet controller, and the PCI interface.

The Reset Device command causes almost the same effect on the SCSI controller that the RST pin does. However, the Reset Device command has no effect on the Ethernet controller or the PCI interface. Also, after the Reset Device command has been issued, the user must issue a NOP command before another command can be executed.

The action of the RST signal or the Reset Device command is called Hard Reset.

The SCSI RST pin is a bidirectional signal on the SCSI bus that resets a portion of the SCSI logic when it is asserted by a device on the SCSI bus. Similarly the Am79C794 can assert the SCSI RST signal to cause all of the other devices on the SCSI bus to reset.

The Reset SCSI command causes the same effect on the SCSI controller that the SCSI RST pin does, except that this command also causes the SCSI RST pin to be asserted so that all other (external) devices on the SCSI bus are also reset. Once a SCSI Reset command has been executed, the SCSI RST signal will remain asserted until a Hard Reset has occurred.

The action of the SCSI RST signal and the Reset SCSI command is called Soft Reset.

In addition there is a third type of reset, called Disconnected Reset, that is caused by certain sequences on the SCSI bus. These three types of reset are described in the following sections.

### Hard Resets: (H)

This reset occurs at power up, when the RST pin is asserted through external hardware, or when the Reset Device command is issued by writing 02h to the SCSI command register at ((B)+0Ch). Hard reset causes all chip functions to halt and resets all internal state machines. It leaves the SCSI block in the disconnected state. It leaves all SCSI registers in their default states.

In addition, if the Hard Reset is caused by the assertion of the RST pin, the following actions occur:

- The Command register in the PCI configuration space is cleared to zero. (No other register in the configuration space is affected.)
- The DMA CCB registers are set to their default values.

### Soft Reset: (S)

This reset occurs either when the SCSI RST pin on the SCSI bus is asserted or when the Reset SCSI Bus command is issued (by writing 03h to the SCSI command register at ((B) = 0ch)).

Soft reset causes the following actions to occur:

- All SCSI bus signals except SCSI RST are released.
- The chip is returned to the Disconnected state.
- An interrupt is generated if bit 6 in Control Register One is enabled.

### Disconnected Reset: (D)

Disconnected reset occurs when either of the following conditions occur:

- The Am79C974 is the Initiator and the SCSI bus moves to a Bus Free state
- The Selection command terminates due to selection time-out

Disconnected reset causes the following actions:

- All SCSI signals except SCSI RST are deasserted.
- The SCSI Command Register is initialized to empty.
- The IS1 and IS0 bits in the Internal State Register, ((B)+18h), are cleared to 0.

Please refer to the *Technical Manual* (PID #18738A) for the default values for all registers.

## Device Commands

The device commands can be broadly divided into two categories, DMA commands and non-DMA commands. DMA commands are those which cause data movement between the host memory and the SCSI bus while non-DMA commands are those that cause data movement between the SCSI FIFO and the SCSI bus. The Most Significant Bit of the command byte differentiates the DMA from the non-DMA commands.

When a DMA command is issued, the contents of the Start Transfer Count Register will be loaded into the Current Transfer Count Register. Data transmission will continue until the Current Transfer Count Register decrements to zero.

Non-DMA commands do not modify the Current Transfer Count Register and are unaffected by the value in the Current Transfer Count Register. For non-DMA commands, the number of bytes transmitted depends solely on the operation in progress.

When non-DMA commands are used, the host computer must use programmed I/O to transfer the data between the SCSI FIFO and the host memory.

**Table 7. Summary of Commands**

| **Command** | **Code (Hex.)** | |
|---|---|---|
| | **Non-DMA Mode** | **DMA Mode** |
| **Initiator Commands** | | |
| Information Transfer | 10 | 90 |
| Initiator Command Complete Steps | 11 | 91 |
| Message Accepted | 12 | – |
| Transfer Pad Bytes | – | 98 |
| Set ATN* | 1A | – |
| Reset ATN* | 1B | – |
| **Idle State Commands** | | |
| Select without ATN Steps | 41 | C1 |
| Select with ATN Steps | 42 | C2 |
| Select with ATN and Stop Steps | 43 | C3 |
| Enable Selection/Reselection* | 44 | C4 |
| Disable Selection/Reselection | 45 | – |
| Select with ATN3 Steps | 46 | C6 |
| **General Commands** | | |
| No Operation* | 00 | 80 |
| Clear FIFO* | 01 | – |
| Reset Device* | 02 | – |
| Reset SCSI Bus** | 03 | – |

***Notes:***

** These commands do not generate interrupt.*

*** An interrupt is generated when SCSI bus reset interrupt reporting is not disabled (see Control Register1/DISR bit6).*

### Command Stacking

The microprocessor may stack commands in the Command Register ((B)+0Ch) since it functions as a two-byte deep FIFO. Non-DMA commands may not be stacked, and commands which transfer data in opposing directions should not be stacked together; otherwise, the results are unpredictable.

If DMA commands are queued together, the Start Transfer Count must be written before the associated command is loaded into the Command Register. Since multiple interrupts can occur when commands are stacked, it is recommended that the ENF bit in Control Register Two (Bit 6) be set in order to latch the SCSI phase bits in the SCSI Status Register ((B)+10h) at the completion of a command. This allows the host to determine the phase of the interrupting command without having to consider phase changes that occurred after the stacked command began execution.

***Note****: Command stacking should only be used during SCSI Data In or Data Out transfers.*

### Invalid Commands

When an illegal command is written to the Am79C974, the Invalid Command Bit (Bit 6, Register (B)+14h) will be set to ‘1’, and an interrupt will be generated to the host. When this happens, the interrupt must be serviced before another command may be written to the Command Register.

An Invalid command is defined as a command written to the Am79C974 that is either not supported, not allowed in the specified mode, or a command that has an unsupported command mode.

The following conditions will also cause an Invalid Command interrupt to occur:

- An Initiator Information Transfer, Transfer Pad, or Command Complete is issued when ACK is still asserted.
- A Selection command is issued with the DMA bit enabled, if the Selection command was previously issued with the DMA enabled.

### Command Window

The window at the point where the Disable Selection/Reselection command (45h/C5h) has been loaded into the Command Register ((B)+0Ch), and before bus-initiated Selection begins, has been eliminated. This prevents a false Successful Operation Interrupt from being generated when the Selection sequence continues to completion after the Disable command has been loaded.

### Initiator Commands

Initiator commands are executed by the device when it is in the Initiator mode. If the device is not in the Initiator mode and an Initiator command is received the device will ignore the command, generate an Invalid Command interrupt and clear the Command Register.

Should the Target disconnect from the SCSI bus by deasserting the BSY signal line while the Am79C974 (Initiator) is waiting for the Target to assert REQ, a Disconnected Interrupt will be issued 1.5 to 3.5 clock cycles following BSY going false.

Upon receipt of the last byte during Message In phase, ACK will remain asserted to prevent the Target from issuing any additional bytes, while the Initiator decides to accept/reject the message. If non-DMA commands are used, the last byte signals the SCSI FIFO is empty. If DMA commands are used, the Current Transfer Count signals the last byte.

A Reset SCSI Bus command (03h/83h) will force the Am79C974 to abort the current operation and disconnect from the bus. If the DISR bit is reset (Bit 6, Control Register One (B)+20h)), the host processor will be interrupted with a SCSI Reset Interrupt before the Am79C974 proceeds to Disconnect.

If parity checking is enabled in the Initiator mode during the Data-in phase and an error is detected, $\overline{\text{ATN}}$ will be asserted for the erroneous byte before deasserting $\overline{\text{ACK}}$.

### Information Transfer Command (Command Code 10h/90h)

The Information Transfer command is used to transfer information bytes over the SCSI bus. This command may be issued during any SCSI Information Transfer phase. Synchronous data transmission requires use of the DMA mode.

The device will continue to transfer information until it is terminated by any one of the following conditions:

- The Target changes the SCSI bus phase before the expected number of bytes are transferred. The Am79C974 clears the Command Register (CMDREG), and generates a Service Request interrupt when the Target asserts $\overline{\text{REQ}}$.
- Transfer has successfully completed. If the phase is Message Out, the Am79C974 deasserts $\overline{\text{ATN}}$ before asserting $\overline{\text{ACK}}$ for the last byte of the message. When the Target asserts $\overline{\text{REQ}}$, a Service Request interrupt is generated.
- In the Message In phase when the device receives the last byte. The Am79C974 keeps the $\overline{\text{ACK}}$ signal asserted and generates a Successful Operation interrupt.

During Synchronous Data transfers the Target may send up to the maximum synchronous offset number of $\overline{\text{REQ}}$ pulses to the Initiator. If it is the Synchronous Data-In phase then the Target sends the data and the $\overline{\text{REQ}}$ pulses. These bytes are stored by the Initiator in the FIFO as they are received.

The Information Transfer Command, when issued during the following SCSI phases and terminated in Synchronous Data phases, is handled as described below:

- Message In/Status Phase – When a phase change to Synchronous Data-In or Synchronous Data-Out is detected by the device, the Command Register is cleared and the DMA interface is disabled to prevent any transfer of data (phase) bytes.
- If the phase change is to Synchronous Data-In and bad parity is detected on the data bytes coming in, it is not reported since the Status Register will report the status of the command just completed. The parity error flag and the $\overline{\text{ATN}}$ signal will be asserted when the next Information Transfer command begins execution.
- Message Out/Command Phase – When a phase change to Synchronous Data-In or Synchronous Data-Out is detected by the device, the Command Register is cleared and the DMA interface is disabled to prevent any transfer of data (phase) bytes.
- If the phase change is to Synchronous Data-In and bad parity is detected on the data bytes coming in, it is not reported since the Status Register will report the status of the command just completed. The parity error flag and the $\overline{\text{ATN}}$ signal will be asserted when the next Information Transfer command begins execution.
- The SCSI FIFO Register will be latched and will remain in that condition until the next command begins execution. The value in the SCSI FIFO Register indicates the number of non-data bytes in the SCSI FIFO when the phase changed to Synchronous Data-In. These bytes are cleared from the FIFO, and only incoming data bytes are retained.
- In the Synchronous Data-Out phase, the threshold counter is incremented as $\overline{\text{REQ}}$ pulses are received. The transfer is completed when the FIFO is empty and the Current Transfer Count Register is ‘0’. The threshold counter will not be ‘0’.
- In the Synchronous Data-In phase, the Current Transfer Count Register is decremented as bytes are read from the SCSI FIFO rather than when the bytes are being written to the SCSI FIFO. The transfer is completed when Current Transfer Count Register is ‘0’. However, the SCSI FIFO may not be empty.

### Initiator Command Complete Steps (Command Code 11h/91h)

The Initiator Command Complete Steps command is normally issued when the SCSI bus is in the Status In phase. One Status byte followed by one Message byte is transferred if this command completes normally. After receiving the message byte the device will keep the $\overline{\text{ACK}}$ signal asserted to allow the Initiator to examine the message and assert the $\overline{\text{ATN}}$ signal if it is unacceptable. The command terminates early if the Target does not switch to the Message In phase or if the Target disconnects from the SCSI bus. This command does not utilize the Internal State Register ((B)+18h).

### Message Accepted Command (Command Code 12h)

The Message Accepted Command is used to release the $\overline{\text{ACK}}$ signal. This command is normally used to complete a Message In handshake. Upon execution of this command the device generates a Service Request interrupt after $\overline{\text{REQ}}$ is asserted by the Target.

After the device has received the last byte of message, it keeps the $\overline{\text{ACK}}$ signal asserted. This allows the device to either accept or reject the message. To accept the message, Message Accepted Command is issued. To reject the message the $\overline{\text{ATN}}$ signal must be asserted (with the help of the Set $\overline{\text{ATN}}$ Command) before issuing the Message Accepted Command. In either case, the Message Accepted Command has to be issued to release the $\overline{\text{ACK}}$ signal.

### Transfer Pad Bytes Command (Command Code 18h/98h)

The Transfer Pad Bytes Command is used to recover from an error condition. This command is similar to the Information Transfer Command, only the information bytes consists of null data. It is used when the Target expects more data bytes than the Initiator has to send. It is also used when the Initiator receives more information than expected from the Target.

When sending data to the SCSI bus, the SCSI FIFO is loaded with null bytes which are sent out to the SCSI bus. Although an actual DMA request is not made, DMA interface must be enabled when pad bytes are transmitted since the Am79C974 uses the Current Transfer Count Register to terminate transmission.

This command terminates under the same conditions as the Information Transfer Command, but the device does not keep the $\overline{\text{ACK}}$ signal asserted during the last byte of the Message In phase. Should this command terminate prematurely due to a Disconnect or a phase change before the Current Transfer Count Register decrements to zero, the SCSI FIFO may contain residual Pad bytes.

### Set $\overline{\text{ATN}}$ Command (Command Code 1Ah)

The Set $\overline{\text{ATN}}$ Command is used to drive the $\overline{\text{ATN}}$ signal active on the SCSI bus. An interrupt is not generated at the end of this command. The $\overline{\text{ATN}}$ signal is deasserted before asserting the $\overline{\text{ACK}}$ signal during the last byte of the Message Out phase.

*Note: The $\overline{\text{ATN}}$ signal is asserted by the device without this command in the following cases:*

- *If any select with $\overline{\text{ATN}}$ command is issued and the arbitration is won.*
- *An Initiator needs the Target's attention to send a message. The $\overline{\text{ATN}}$ signal is asserted before deasserting the $\overline{\text{ACK}}$ signal.*

### Reset $\overline{\text{ATN}}$ Command (Command Code 1Bh)

The Reset $\overline{\text{ATN}}$ Command is used to deassert the $\overline{\text{ATN}}$ signal on the SCSI bus. An interrupt is not generated at the end of this command. This command is used only when interfacing with devices that do not support the Common Command Set (CCS). These older devices do not deassert their $\overline{\text{ATN}}$ signal automatically on the last byte of the Message Out phase. This device does deassert its $\overline{\text{ATN}}$ signal automatically on the last byte of the Message Out phase.

### Idle State Commands

The Idle State Commands can be issued to the device only when the device is disconnected from the SCSI bus. If these commands are issued to the device when it is logically connected to the SCSI bus, the commands are ignored, an Invalid Command interrupt is generated, and the Command Register (CMDREG) is cleared.

### *Select Without $\overline{\text{ATN}}$ Steps Command (Command Code 41h/C1h)*

The Select without $\overline{\text{ATN}}$ Steps Command is used by the Initiator to select a Target. When this command is issued, the device arbitrates for the control of the SCSI bus. When the device wins arbitration, it selects the Target device and transfers the Command Descriptor Block (CDB). Before issuing this command the SCSI Timeout Register (STIMREG), Control Register One (CNTLREG1), and the SCSI Destination ID Register (SDIDREG) must be set to the proper values. If DMA is enabled, the Start Transfer Count Register (STCREG) must be set to the total length of the command. If DMA is not enabled, the data must be loaded into the FIFO before issuing this command. This command will be terminated early if the SCSI Timeout Register times out, if the Target does not go to the Command Phase following the Selection Phase, or if the Target exits the Command Phase prematurely. A Successful Operation interrupt will be generated following normal command execution.

### *Select With $\overline{\text{ATN}}$ Steps Command (Command Code 42h/C2h)*

The Select with $\overline{\text{ATN}}$ Steps Command is used by the Initiator to select a Target. When this command is issued, the device arbitrates for the control of the SCSI bus. When the device wins arbitration, it selects the Target device with the $\overline{\text{ATN}}$ signal asserted and transfers the Command Descriptor Block (CDB) and a one byte message. Before issuing this command the SCSI Timeout Register (STIMREG), Control Register One (CNTLREG1) and the SCSI Destination ID Register (SDIDREG) must be set to the proper values. If DMA is enabled, the Start Transfer Count Register (STCREG) must be set to the total length of the command and message. If DMA is not enabled, the data must be loaded into the FIFO before issuing this command. This command will be terminated early in the following situations:

- The SCSI Timeout Register times out
- The Target does not go to the Message Out Phase following the Selection Phase
- The Target exits the Message Phase early
- The Target does not go to the Command Phase following the Message Out Phase
- The Target exits the Command Phase early

A Successful Operation/Service Request interrupt is generated when this command is completed successfully.

### *Select With $\overline{\text{ATN}}$ and Stop Steps Command (Command Code 43h/C3h)*

The Select with $\overline{\text{ATN}}$ and Stop Steps Command is used by the Initiator to send messages with lengths other than 1 or 3 bytes. When this command is issued, the device executes the Selection process, transfers the first message byte, then STOPS the sequence. $\overline{\text{ATN}}$ is not deasserted at this time, allowing the Initiator to send additional message bytes after the ID message. To send these additional bytes, the Initiator must write the transfer counter with the number of bytes which will follow, then issue a Transfer Information Command. (Note: the Target is still in the Message Out phase when this command is issued). $\overline{\text{ATN}}$ will remain asserted until the Current Transfer Count Register decrements to zero.

The SCSI Timeout Register (STIMREG), Control Register One (CNTLREG1), and the SCSI Destination ID Register (SDIDREG) must be set to the proper values before the Initiator issues this command. This command will be terminated early if the STIMREG times out or if the Target does not go to the Message Out Phase following the Selection Phase.

### *Enable Selection/Reselection Command (Command Code 44H/C4H)*

The Enable Selection/Reselection Command is used to respond to a bus-initiated Selection or Reselection. Upon disconnecting from the bus the Selection/Reselection circuit is automatically disabled by the device. This circuit must be enabled for the Am79C974 to respond to subsequent reselection attempts and the Enable Selection/Reselection Command is issued to do that. This command is normally issued within 250 ms (select/reselect timeout) after the device disconnects from the bus. If DMA is enabled, the device loads the received data to the buffer memory. If the DMA is disabled, the received data stays in the FIFO.

### *Disable Selection/Reselection Command (Command Code 45H)*

The Disable Selection/Reselection Command is used by the Target to disable response to a bus-initiated Reselection. When this command is issued before a bus-initiated Selection or Reselection is in progress, it resets the internal state bits previously set by the Enable Selection/Reselection Command. The device also generates a Successful Operation interrupt to the processor. If however, this command is issued after a bus-initiated Selection/Reselection has begun, this command and all incoming commands are ignored since the Command Register (CMDREG) is held reset. The Am79C974 also generates a Selected or Reselected interrupt when the sequence is complete.

### *Select With $\overline{\text{ATN}}$3 Steps Command (Command Code 46h/C6h)*

The Select with $\overline{\text{ATN}}$3 Steps Command is used by the Initiator to select a Target. This command is similar to the Select with $\overline{\text{ATN}}$ Steps Command, except that it sends exactly three message bytes. When this command is issued the Am79C974 arbitrates for control of the SCSI bus. When the device wins arbitration, it selects the Target device with the $\overline{\text{ATN}}$ signal asserted and transfers the Command Descriptor Block (CDB) and three message bytes. Before issuing this command the SCSI Timeout Register (STIMREG), Control Register One (CNTLREG1), and the SCSI Destination ID Register (SDIDREG) must be set to the proper values. If DMA is enabled, the Start Transfer Count Register (STCREG) must be set to the total length of the command. If DMA is not enabled, the data must be loaded into the FIFO before issuing this command. This command will be terminated early in the following situations:

- The SCSI Timeout Register times out
- The Target does not go to the Message Out Phase following the Selection Phase
- The Target removes Command Phase early
- The Target does not go to the Command Phase following the Message Out Phase
- The Target exits the Command Out Phase early

A Successful Operation/Service Request interrupt is generated when this command is executed successfully.

## General Commands

### *No Operation Command (Command Code 00h/80h)*

The No Operation Command administers no operation, therefore an interrupt is not generated upon completion. This command is issued following the Reset Device Command to clear the Command Register (CMDREG). A No Operation Command in the DMA mode may be used to verify the contents of the Start Transfer Count Register (STCREG). After the STCREG is loaded with the transfer count and a DMA No Operation Command is issued, reading the Current Transfer Count Register (CTCREG) returns the transfer count value.

### *Clear FIFO Command (Command Code 01h)*

The Clear FIFO Command is used to initialize the SCSI FIFO to the empty condition. The Current FIFO Register (CFISREG) reflects the empty FIFO status and the bottom of the FIFO is set to zero. No interrupt is generated at the end of this command.

### *Reset Device Command (Command Code 02h)*

The Reset Device Command immediately stops any device operation and resets all the functions of the device. Additionally, it returns the device to the disconnected state and it generates a hard reset. The Reset Device Command remains on the top of the Command Register FIFO holding the device in the reset state until the No Operation Command is loaded. Once loaded, the No Operation command serves to re-enable the Command Register.

### *Reset SCSI Bus Command (Command Code 03h)*

The Reset SCSI Bus Command forces the $\overline{\text{SCSI RST}}$ signal active for a period of 25 ms, and drives the chip to the Disconnected state. An interrupt is not generated upon command completion, however, if bit 6 is set to '0' in Control Register One (CNTLREG1), a SCSI Reset interrupt will be issued.

## SCSI Power Management Features

As a leader in low-voltage technology, AMD has incorporated power-saving features into the Am79C974. Through hardware and software, the Am79C974 can be powered down to reduce consumption during chip inactivity.

### SCSI Activity Pin

The SCSI Bus activity is reflected by the $\overline{\text{BUSY}}$ output line. This signal, when active, indicates that the SCSI Bus is in use, therefore the Am79C974 should not be powered down. This signal is the logical equivalent to the SCSI bus signal $\overline{\text{BSY}}$, however, it is not physically connected to the $\overline{\text{BSY}}$ signal on the bus. To correctly identify the Bus Free State on the SCSI bus, this $\overline{\text{BUSY}}$ output line must be inactive for at least 250 ms (Selection Timeout period).

### *Reduced Power Mode*

When there is no activity on the SCSI Bus, and there are no pending commands, the Am79C974 may be powered down by turning off the input buffers on the SCSI Bus lines (set bit 5 in Control Register Four (B)+34h). For further power reduction, the internal registers may be programmed to a predetermined state, and the input clock disconnected via external logic.

### Power Down Pin (PWDN Pin)

When PWDN is driven active, it sets the PWDN bit in the Status Register (Bit 0, DMA Status Register (B)+54h), signaling the Am79C974 that the host would like to power down the SCSI interface. An interrupt is generated when this bit is first set.

### *Software Disk Spin-Down*

Incorporated into the SCSI ROM BIOS and certain device driver's of AMD's software solution is a module which physically spins down SCSI fixed disks when the host system elects to enact power management on the SCSI system. The software module is activated upon the ROM BIOS and/or other device driver's receipt of an interrupt caused by the PWDN pin being driven active. Upon receipt of this interrupt, the current software process is suspended and software control is given to the power management module.

When the power management module is activated, it checks the status of all SCSI fixed disks on the system under its control. The SCSI fixed disks that are idle are issued a command to spin down their media. All fixed disks that are active at the time will be scheduled for spin down upon completion of their pending commands. Once the BIOS and/or drivers have detected the completion of each fixed disk's final pending commands, they are issued the command to spin down as well.

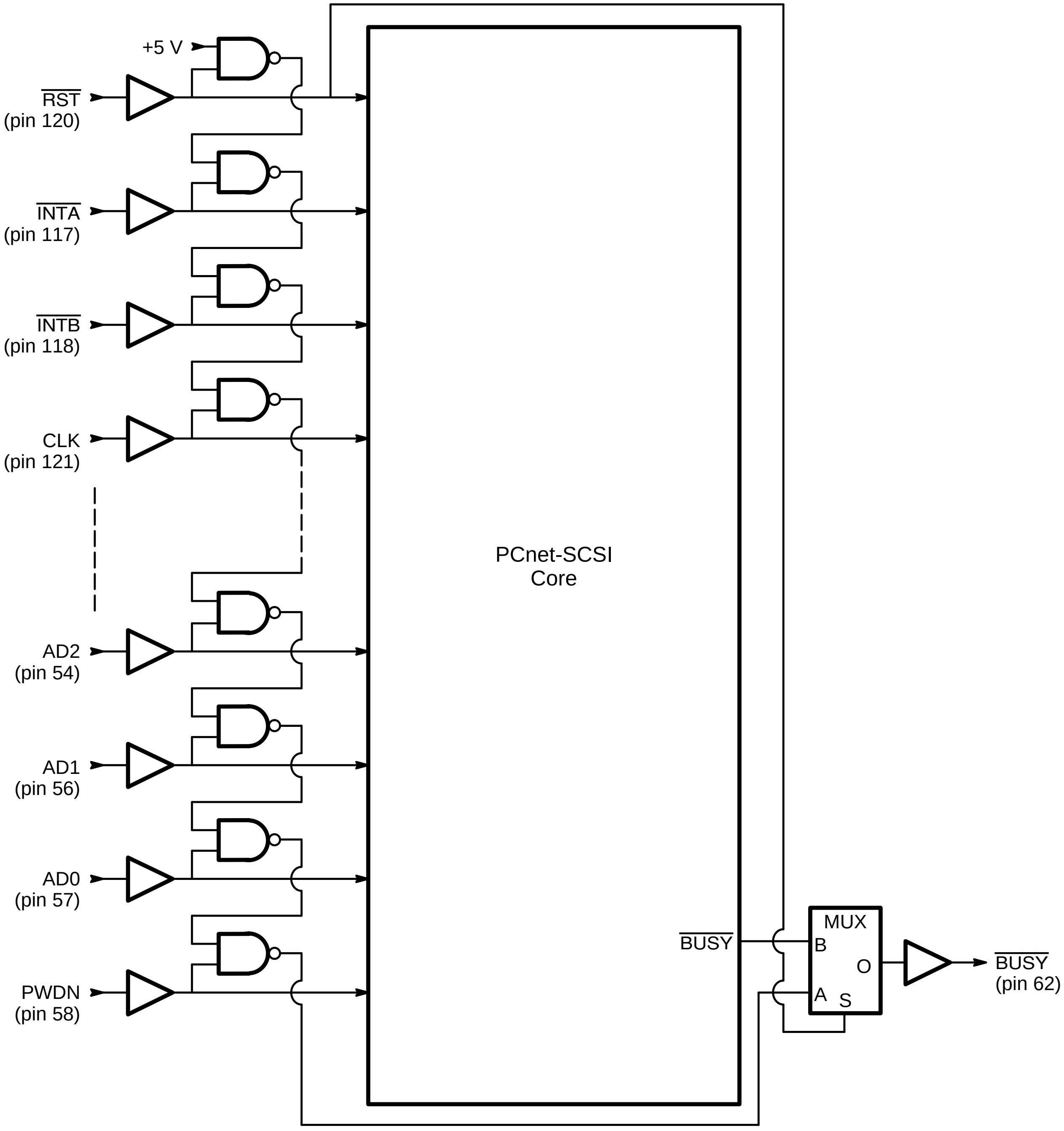
To spin down the disk drives, the power management module issues a SCSI command (1B– Start/Stop unit). When the command is received by the drive, it spins down and waits in an idle state. For multiple drives on the SCSI bus, the power management driver spins down each drive individually. The drives remain in the idle state until the BIOS and/or driver receives a command for the particular fixed disk. Once this occurs, the drive is issued the command to spin up. When the drive has completely spun up and is ready, the pending command is issued to the drive. Only the particular fixed disk issued the command is instructed to spin up. All other drives will remain in the spun down state until a command is issued to them.

***Note****: This sequence does not turn off the input buffers as described in the previous section.*

### NAND Tree Testing

The Am79C974 controller provides a NAND tree test mode to allow checking connectivity to the device on a printed circuit board. The NAND tree is built on all PCI bus signals (see Figure 27 and Table 8).

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


+5 V
$\overline{\text{RST}}$ (pin 120)
$\overline{\text{INTA}}$ (pin 117)
$\overline{\text{INTB}}$ (pin 118)
CLK (pin 121)
AD2 (pin 54)
AD1 (pin 56)
AD0 (pin 57)
PWDN (pin 58)
PCnet-SCSI Core
$\overline{\text{BUSY}}$
MUX
B
A
S
O
$\overline{\text{BUSY}}$ (pin 62)

18681A-35

**Figure 27. Am79C974 NAND Tree Test Structure**

As shown in Figure 27, Pin 120 ($\overline{\text{RST}}$) is the first input to the NAND tree. Pin 117 ($\overline{\text{INTA}}$) is the second input to the NAND tree, followed by pin 118 ($\overline{\text{INTB}}$). All other PCI bus signals follow, counterclockwise, with pin 58 (PWDN) being the last. Ethernet and SCSI specific pins and all power supply pins are not part of the NAND tree. Table 8 shows the complete list of pins connected to the NAND tree.

**Table 8. NAND Tree Configuration**

| NAND Tree Input # | Pin # | Name | NAND Tree Input # | Pin # | Name | NAND Tree Input # | Pin # | Name |
|---|---|---|---|---|---|---|---|---|
| 1 | 120 | $\overline{\text{RST}}$ | 20 | 12 | AD23 | 39 | 36 | AD15 |
| 2 | 117 | $\overline{\text{INTA}}$ | 21 | 13 | AD22 | 40 | 38 | AD14 |
| 3 | 118 | $\overline{\text{INTB}}$ | 22 | 15 | AD21 | 41 | 39 | AD13 |
| 4 | 121 | CLK | 23 | 16 | AD20 | 42 | 40 | AD12 |
| 5 | 123 | $\overline{\text{GNTB}}$ | 24 | 18 | AD19 | 43 | 41 | AD11 |
| 6 | 124 | $\overline{\text{GNTA}}$ | 25 | 19 | AD18 | 44 | 42 | AD10 |
| 7 | 126 | $\overline{\text{REQB}}$ | 26 | 21 | AD17 | 45 | 44 | AD9 |
| 8 | 127 | $\overline{\text{REQA}}$ | 27 | 22 | AD16 | 46 | 45 | AD8 |
| 9 | 128 | AD31 | 28 | 23 | C/$\overline{\text{BE}}$2 | 47 | 47 | C/$\overline{\text{BE}}$0 |
| 10 | 129 | AD30 | 29 | 24 | $\overline{\text{FRAME}}$ | 48 | 48 | AD7 |
| 11 | 131 | AD29 | 30 | 25 | $\overline{\text{IRDY}}$ | 49 | 49 | AD6 |
| 12 | 132 | AD28 | 31 | 26 | $\overline{\text{TRDY}}$ | 50 | 51 | AD5 |
| 13 | 2 | AD27 | 32 | 27 | $\overline{\text{DEVSEL}}$ | 51 | 52 | AD4 |
| 14 | 3 | AD26 | 33 | 28 | $\overline{\text{STOP}}$ | 52 | 53 | AD3 |
| 15 | 5 | AD25 | 34 | 29 | $\overline{\text{LOCK}}$ | 53 | 54 | AD2 |
| 16 | 6 | AD24 | 35 | 31 | $\overline{\text{PERR}}$ | 54 | 56 | AD1 |
| 17 | 7 | C/$\overline{\text{BE}}$3 | 36 | 32 | $\overline{\text{SERR}}$ | 55 | 57 | AD0 |
| 18 | 9 | IDSELA | 37 | 34 | PAR | 56 | 58 | PWDN |
| 19 | 10 | IDSELB | 38 | 35 | C/$\overline{\text{BE}}$1 | | | |

RST must be asserted low to start a NAND tree test sequence. Initially, all NAND tree inputs except RST should be driven high. This will result in a high output at the BUSY pin. If the NAND tree inputs are driven low in the same order as they are connected to build the NAND tree, BUSY will toggle every time an additional input is driven low. BUSY will change to a ZERO, when INTA is driven low and all other NAND tree inputs stay high. BUSY will toggle back to high, when CLK is additionally driven low. BUSY will continue toggling as long as the NAND tree inputs are toggling in the sequence shown in Figure 28. BUSY will be high, when all NAND tree inputs are driven low.

When testing is complete, deassert RST to exit this test mode.

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

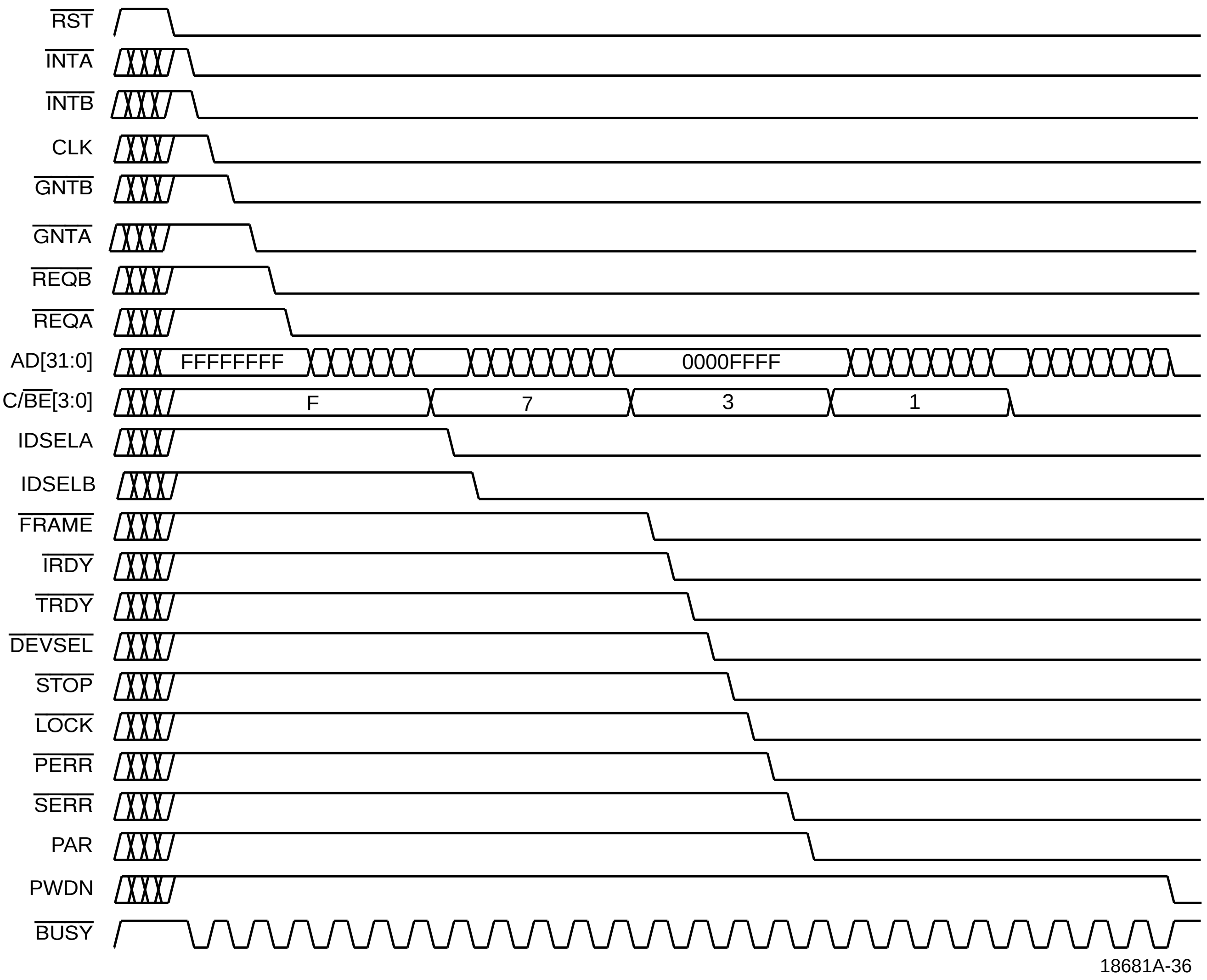


**Figure 28. NAND Tree Waveform**

## ABSOLUTE MAXIMUM RATINGS

Storage Temperature . . . . . . . . . . . –65°C to +150°C

Ambient Temperature
Under Bias . . . . . . . . . . . . . . . . . . . –55°C to +125°C

Supply Voltage to $AV_{SS}$ or $V_{SSB}$ or $V_{SSBS}$ or $DV_{SS}$
($AV_{DD}$, $V_{DD}$, $V_{DDB}$, $V_{DDBS}$, $DV_{DD}$) . . . –0.3 V to +6.0 V

*Stresses above those listed under Absolute Maximum Ratings may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.*

## OPERATING RANGES

Temperature ($T_A$) . . . . . . . . . . . . . . . . . 0°C to +70°C

Supply Voltages
($AV_{DD}$, $V_{DD}$, $V_{DDB}$, $V_{DDBS}$, $DV_{DD}$) . . . . . . . . . +5 V ± 5%

All Inputs within the Range:
$AV_{SS} - 0.5\ V \le V_{IN} \le AV_{DD} + 0.5\ V$,
or $V_{SS} - 0.5\ V \le V_{IN} \le V_{DD} + 0.5\ V$,
or $V_{SSB} - 0.5\ V \le V_{IN} \le V_{DDB} + 0.5\ V$
or $V_{SSBS} - 0.5\ V \le V_{IN} \le V_{DDBS} + 0.5\ V$
or $DV_{SS} - 0.5\ V \le V_{IN} \le DV_{DD} + 0.5\ V$

*Operating ranges define those limits between which the functionality of the device is guaranteed.*

## DC CHARACTERISTICS over COMMERCIAL operating range unless otherwise specified

### PCI and Board Interface

| Parameter Symbol | Parameter Description | Test Conditions | | Min | Max | Unit |
|---|---|---|---|---|---|---|
| **Digital Input Voltage** | | | | | | |
| $V_{IL}$ | Input LOW Voltage | | | | 0.8 | V |
| $V_{IH}$ | Input HIGH Voltage | | | 2.0 | | V |
| **Digital Output Voltage** | | | | | | |
| $V_{OL}$ | Output LOW Voltage | $I_{OL1}$ = 3 mA<br>$I_{OL2}$ = 6 mA | | | 0.55 | V |
| | | $I_{OL3}$ = 12 mA<br>(Note 1) | | | 0.4 | V |
| $V_{OH}$ | Output HIGH Voltage (Note 2) | $I_{OH}$ = –2 mA (Note 5) | | 2.4 | | V |
| **Digital Input Leakage Current** | | | | | | |
| $I_{IL}$ | Input Low Leakage Current (Note 3) | $V_{IN}$ = 0 | | –10 | 10 | µA |
| $I_{IH}$ | Input High Leakage Current (Note 3) | $V_{IN}$ = $V_{DD}$, $V_{DDB}$ | | –10 | +10 | µA |
| **Digital Output Leakage Current** | | | | | | |
| $I_{OZL}$ | Output Low Leakage Current (Note 4) | $V_{OUT}$ = 0.4 V | | –10 | +10 | µA |
| $I_{OZH}$ | Output High Leakage Current (Note 4) | $V_{OUT}$ = $V_{DD}$, $V_{DDB}$ | | –10 | +10 | µA |
| **Crystal Input Current** | | | | | | |
| $V_{ILX}$ | XTAL1 Input LOW Voltage Threshold | $V_{IN}$ = External Clock | | –0.5 | 0.8 | V |
| $V_{IHX}$ | XTAL1 Input HIGH Voltage Threshold | $V_{IN}$ = External Clock | | $V_{DD}$ – 0.8 | $V_{DD}$ + 0.5 | V |
| $I_{ILX}$ | XTAL1 Input LOW Current | $V_{IN}$ = External Clock | Active | –120 | 0 | µA |
| | | $V_{IN}$ = $V_{SS}$ | Sleep | –10 | +10 | µA |
| $I_{IHX}$ | XTAL1 Input HIGH Current | $V_{IN}$ = External Clock | Active | 0 | 120 | µA |
| | | $V_{IN}$ = $V_{DD}$ | Sleep | | 400 | µA |

## DC CHARACTERISTICS (continued)

### Attachment Unit Interface

| Parameter Symbol | Parameter Description | Test Conditions | Min | Max | Unit |
|---|---|---|---|---|---|
| **Attachment Unit Interface (AUI)** | | | | | |
| $I_{IAXD}$ | Input Current at DI+ and DI– | $–1 V < V_{IN} < AV_{DD}+0.5V$ | –500 | +500 | µA |
| $I_{IAXC}$ | Input Current at CI+ and CI– | $–1 V < V_{IN} < AV_{DD}+0.5V$ | –500 | +500 | µA |
| $V_{AOD}$ | Differential Output Voltage \|(DO+)–(DO–)\| | $R_L = 78\ \Omega$ | 630 | 1200 | mV |
| $V_{AODOFF}$ | Transmit Differential Output Idle Voltage | $R_L = 78\ \Omega$ (Note 9) | –40 | 40 | mV |
| $I_{AODOFF}$ | Transmit Differential Output Idle Current | $R_L = 78\ \Omega$ (Note 8) | –1 | 1 | mA |
| $V_{CMT}$ | Transmit Output Common Mode Voltage | $R_L = 78\ \Omega$ | 2.5 | $A_{VDD}$ | V |
| $V_{ODI}$ | DO± Transmit Differential Output Voltage Imbalance | $R_L = 78\ \Omega$ (Note 7) | | 25 | mV |
| $V_{ATH}$ | Receive Data Differential Input Threshold | | –35 | 35 | mV |
| $V_{ASQ}$ | DI± and CI± Differential Input Threshold (Squelch) | | –275 | –160 | mV |
| $V_{IRDVD}$ | DI± and CI± Differential Mode Input Voltage Range | | –1.5 | 1.5 | V |
| $V_{ICM}$ | DI± and CI± Input Bias Voltage | $I_{IN} = 0$ mA | $AV_{DD}–3.0$ | $AV_{DD}–1.0$ | V |
| $V_{OPD}$ | DO± Undershoot Voltage at ZERO Differential on Transmit Return to ZERO (ETD) | (Note 9) | | –100 | mV |
| **Twisted Pair Interface (10BASE-T)** | | | | | |
| $I_{IRXD}$ | Input Current at RXD± | $AV_{DD} < V_{IN} < AV_{DD}$ | –500 | 500 | µA |
| $R_{RXD}$ | RXD± Differential Input Resistance | | 10 | | KΩ |
| $V_{TIVB}$ | RXD±, RXD– Open Circuit Input Voltage (Bias) | $I_{IN} = 0$ mA | $AV_{DD}–3.0$ | $AV_{DD}–1.5$ | V |
| $V_{TIDV}$ | Differential Mode Input Voltage Range (RXD±) | $AV_{DD} = 5.0$ V | –3.1 | 3.1 | V |
| $V_{TSQ+}$ | RXD Positive Squelch Threshold (peak) | Sinusoid, 5 MHz ≤ f ≤ 10 MHz | 300 | 520 | mV |
| $V_{TSQ-}$ | RXD Negative Squelch Threshold (peak) | Sinusoid, 5 MHz ≤ f ≤ 10 MHz | –520 | –300 | mV |
| $V_{THS+}$ | RXD Post-Squelch Positive Threshold (peak) | Sinusoid, 5 MHz ≤ f ≤ 10 MHz | 150 | 293 | mV |
| $V_{THS-}$ | RXD Post-Squelch Negative Threshold (peak) | Sinusoid, 5 MHz ≤ f ≤ 10 MHz | –293 | –150 | mV |
| $V_{LTSQ+}$ | RXD Positive Squelch Threshold (peak) | LRT = LOW | 180 | 312 | mV |
| $V_{LTSQ-}$ | RXD Negative Squelch Threshold (peak) | LRT = LOW | –312 | –180 | mV |
| $V_{LTHS+}$ | RXD Post-Squelch Positive Threshold (peak) | LRT = LOW | 90 | 176 | mV |
| $V_{LTHS-}$ | RXD Post-Squelch Negative Threshold (peak) | LRT = LOW | –176 | –90 | mV |
| $V_{RXDTH}$ | RXD Switching Threshold | (Note 4) | –35 | 35 | mV |
| $V_{TXH}$ | TXD± and TXP± Output HIGH Voltage | $V_{SS} = 0$ V | $V_{DD}–0.6$ | $V_{DD}$ | V |
| $V_{TXL}$ | TXD± and TXP± Output LOW Voltage | $V_{DD} = 5$ V | $V_{SS}$ | $V_{SS}+0.6$ | V |
| $V_{TXI}$ | TXD± and TXP± Differential Output Voltage Imbalance | | –40 | 40 | mV |
| $V_{TX}OFF$ | TXD± and TXP± Idle Output Voltage | | | 40 | mV |
| $R_{TX}$ | TXD± , TXP± Differential Driver Output Impedance | (Note 4) | 40 | 80 | Ω |

## DC CHARACTERISTICS (continued)

### SCSI Interface

| Parameter Symbol | Parameter Description | Test Conditions | Min | Max | Unit |
|---|---|---|---|---|---|
| **SCSI and PWDN** | | | | | |
| $V_{IH}$ | Input High Voltage<br>All SCSI Inputs, PWDN | | 2.0 | $V_{DD}$ + 0.5 | V |
| $V_{IL}$ | Input Low Voltage<br>All SCSI Inputs, PWDN | | $V_{SS}$ – 0.5 | 0.8 | V |
| $V_{IHST}$ | Input Hysteresis (Note 6)<br>All SCSI Inputs | 4.75 V < $V_{DD}$ < 5.25 V | 300 | | mV |
| $V_{OH}$ | Output High Voltage (Note 2) | $I_{OH}$ = –2 mA (Note 5) | 2.4 | $V_{DD}$ | V |
| $V_{OL}$ | SCSI Output Low Voltage<br>$\overline{\text{ATN}}$, $\overline{\text{BSY}}$, $\overline{\text{SEL}}$,<br>$\overline{\text{SCSI RST}}$, $\overline{\text{ACK}}$, $\overline{\text{SD}}$ [7:0], $\overline{\text{SDP}}$ | $I_{OL}$ = 48 mA | $V_{SS}$ | 0.5 | V |
| $I_{IL}$ | Input Low Leakage<br>All SCSI Inputs, PWDN | 0.0 V < $V_{IN}$ < 2.7 V | –10 | +10 | µA |
| $I_{IH}$ | Input High Leakage<br>All SCSI Inputs, PWDN | 2.7 V < $V_{IN}$ < $V_{DD}$ | –10 | +10 | µA |
| $I_{OZ}$ | High Impedance Leakage<br>All I/O Pins | 0.4 V < $V_{OUT}$ < $V_{DD}$ | –10 | +10 | µA |
| **Power Supply Current** | | | | | |
| $I_{DD}$ | Active Power Supply Current | XTAL1 = 20 MHz, CLK = 33 MHz | | 150 | mA |
| $I_{DDCOMA}$ | Sleep Mode Power Supply Current | $\overline{\text{SLEEP}}$ Active | | TBD | µA |
| $I_{DDSNOOZE}$ | Auto Wake Mode Power Supply Current | Awake Bit Set Active | | TBD | µA |

***Notes:***

1. *$I_{OL1}$ applies to AD[31:00], C/$\overline{\text{BE}}$[3:0], PAR, and* $\overline{\text{REQ}}$
   *$I_{OL2}$ applies to* $\overline{\text{FRAME}}$, $\overline{\text{IRDY}}$, $\overline{\text{TRDY}}$, $\overline{\text{DEVSEL}}$, $\overline{\text{STOP}}$, $\overline{\text{SERR}}$, $\overline{\text{PERR}}$, *and* $\overline{\text{LOCK}}$
   *$I_{OL3}$ applies to EESK/$\overline{\text{LED1}}$, EEDO/$\overline{\text{LED3}}$, and EEDI/$\overline{\text{LNKST}}$*
2. *$V_{OH}$ does not apply to open-drain output pins.*
3. *$I_{IL}$ and $I_{IH}$ apply to all input pins except XTAL1.*
4. *$I_{OZL}$ and $I_{OZH}$ apply to all three-state output pins and bi-directional pins.*
5. *Outputs are CMOS and will be driven to rail if the load is not resistive.*
6. *These parameters are not 100% tested, but are evaluated at initial characterization and at any time the design is modified where these parameters may be affected.*
7. *Tested, but to values in excess of limits. Test accuracy not sufficient to allow screening guard bands.*
8. *Correlated to other tested parameters – not tested directly.*
9. *Test not implemented to data sheet specification.*

## DC CHARACTERISTICS (continued)

## Capacitance, ESD, and Latch Up

| Parameter Symbol | Parameter Description | Pin Names | Test Conditions | Min | Max | Unit |
|---|---|---|---|---|---|---|
| **Pin Capacitance (Note 1) ($V_{CC}$ = 5.0 V, $T_A$ = 25∝C, f = 1.0 MHz)** | | | | | | |
| $C_{IN}$ | Input Pins | All SCSI Inputs<br>All Ethernet Inputs<br>PWDN<br>All PCI Inputs except IDSEL | $V_{IN}$ = 0 V | | 10 | pF |
| | | IDSEL | $V_{IN}$ = 0 V | | 8 | pF |
| $C_{I/O}$ | I/O or Output Pins | All SCSI, Ethernet, PCI Output and I/O Pins, $\overline{\text{BUSY}}$ | $V_{I/O}$ = 0 V | | 12 | pF |
| $C_{CLK}$ | Clock Pins | CLK (PCI)<br>SCSI CLK | $V_{IN}$ = 0 V | | 12 | pF |
| **ESD (Note 1)** | | | | | | |
| | Input Static Discharge | Pin-to-Pin | Human Body Model: 100 pF at 1.5 KΩ | 2K | | V |
| **Latchup (Note 1)** | | | | | | |
| $I_{LU}$ | Latchup Current | All I/O | $V_{LU}$ ≤ 10 V | –100 | +100 | mA |

***Note:***

*1. These parameters are not 100% tested, but are evaluated at initial characterization and at any time the design is modified where these parameters may be affected.*

## AC SWITCHING CHARACTERISTICS over operating range unless otherwise specified

## PCI Bus Interface and Board Interface

| Parameter Symbol | Parameter Description | Test Conditions | Min | Max | Unit |
|---|---|---|---|---|---|
| **Clock Timing** | | | | | |
| | CLK Frequency | | 0 | 33 | MHz |
| $t_{CYC}$ | CLK Period | @ 1.5 V | 30 | ∞ | ns |
| $t_{HIGH}$ | CLK High Time | @ 2.0 V | 12 | | ns |
| $t_{LOW}$ | CLK Low Time | @ 0.8 V | 12 | | ns |
| $t_{FALL}$ | CLK Fall Time | over 2 V p-p | 1 | 4 | V/ns |
| $t_{RISE}$ | CLK Rise Time | over 2 V p-p | 1 | 4 | V/ns |
| **Output and Float Delay Timing** | | | | | |
| $t_{VAL}$ | AD[31:00], C/$\overline{\text{BE}}$[3:0], PAR, $\overline{\text{FRAME}}$, $\overline{\text{IRDY}}$, $\overline{\text{TRDY}}$, $\overline{\text{STOP}}$, $\overline{\text{LOCK}}$, $\overline{\text{DEVSEL}}$, $\overline{\text{PERR}}$, $\overline{\text{SERR}}$ Valid Delay | | 2 | 11 | ns |
| $t_{VAL}$ ($\overline{\text{REQ}}$) | $\overline{\text{REQ}}$ Valid Delay | | 1 | 12 | ns |
| $f_{EESK}$ | EESK Frequency | (See note below) | | 650 | KHz |
| $t_{VAL}$ (EEDI) | EEDI Valid Output Delay from EESK | (See note below) | 100 | 400 | ns |
| $t_{VAL}$ (EESK) | EECS Valid Output Delay from EESK | (See note below) | 0 | 400 | ns |
| $t_{ON}$ | AD[31:00], C/$\overline{\text{BE}}$[3:0], PAR, $\overline{\text{FRAME}}$, $\overline{\text{IRDY}}$, $\overline{\text{TRDY}}$, $\overline{\text{STOP}}$, $\overline{\text{LOCK}}$, $\overline{\text{DEVSEL}}$ Active Delay | | 2 | 11 | ns |
| $t_{OFF}$ | AD[31:00], C/$\overline{\text{BE}}$[3:0], PAR, $\overline{\text{FRAME}}$, $\overline{\text{IRDY}}$, $\overline{\text{TRDY}}$, $\overline{\text{STOP}}$, $\overline{\text{LOCK}}$, $\overline{\text{DEVSEL}}$ Float Delay | | | 28 | ns |
| **Setup and Hold Timing** | | | | | |
| $t_{SU}$ | AD[31:00], C/$\overline{\text{BE}}$[3:0], PAR, $\overline{\text{FRAME}}$, $\overline{\text{IRDY}}$, $\overline{\text{TRDY}}$, $\overline{\text{STOP}}$, $\overline{\text{LOCK}}$, $\overline{\text{DEVSEL}}$, IDSEL Setup Time | | 7 | | ns |
| $t_{H}$ | AD[31:00], C/$\overline{\text{BE}}$[3:0], PAR, $\overline{\text{FRAME}}$, $\overline{\text{IRDY}}$, $\overline{\text{TRDY}}$, $\overline{\text{STOP}}$, $\overline{\text{LOCK}}$, $\overline{\text{DEVSEL}}$, IDSEL Hold Time | | 0 | | ns |
| $t_{SU}$ ($\overline{\text{GNT}}$) | $\overline{\text{GNT}}$ Setup Time | | 10 | | ns |
| $t_{H}$ ($\overline{\text{GNT}}$) | $\overline{\text{GNT}}$ Hold Time | | 0 | | ns |
| $t_{SU}$ (EEDO) | EEDO Setup Time to EESK | (See note below) | 50 | | ns |
| $t_{H}$ (EEDO) | EEDO Hold Time from EESK | (See note below) | 0 | | ns |

***Note:***

*Parameter value is given for automatic EEPROM read operation. When EEPROM port (BCR19) is used to access the EEPROM, software is responsible for meeting EEPROM timing requirements.*

## AC SWITCHING CHARACTERISTICS

### 10BASE-T Interface

| Parameter Symbol | Parameter Description | Test Conditions | Min | Max | Unit |
|---|---|---|---|---|---|
| **Transmit Timing** | | | | | |
| $t_{TETD}$ | Transmit Start of Idle | | 250 | 350 | ns |
| $t_{TR}$ | Transmitter rise time | (10% to 90%) | | 5.5 | ns |
| $t_{TF}$ | Transmitter fall time | (90% to 10%) | | 5.5 | ns |
| $t_{TM}$ | Transmitter rise and fall time mismatch | ($t_{TM} = \| t_{TR} – t_{TF}\|$) | | 1 | ns |
| $t_{PERLP}$ | Idle Signal Period | | 8 | 24 | ms |
| $t_{PWLP}$ | Idle Link Pulse Width | (See note below) | 75 | 120 | ns |
| $t_{PWPLP}$ | Predistortion Idle Link Pulse Width | (See note below) | 45 | 55 | ns |
| $t_{JA}$ | Transmit jabber activation time | | 20 | 150 | ms |
| $t_{JR}$ | Transmit jabber reset time | | 250 | 750 | ms |
| $t_{JREC}$ | Transmit jabber recovery time (minimum time gap between transmitted frames to prevent jabber activation) | | 1.0 | | µs |
| **Receiving Timing** | | | | | |
| $t_{PWNRD}$ | RXD pulse width not to turn off internal carrier sense | $V_{IN} > V_{THS}$ (min) | 136 | | ns |
| $t_{PWROFF}$ | RXD pulse width to turn off | $V_{IN} > V_{THS}$ (min) | | 200 | ns |
| $t_{RETD}$ | Receive Start of Idle | | 200 | | ns |

***Note:***

*Not tested; parameter guaranteed by characterization.*

## AC SWITCHING CHARACTERISTICS

### Attachment Unit Interface

| Parameter Symbol | Parameter Description | Test Conditions | Min | Max | Unit |
|---|---|---|---|---|---|
| **AUI Port** | | | | | |
| $t_{DOTR}$ | DO+, DO– Rise Time (10% to 90%) | | 2.5 | 5.0 | ns |
| $t_{DOTF}$ | DO+, DO– Fall Time (10% to 90%) | | 2.5 | 5.0 | ns |
| $t_{DORM}$ | DO+, DO– Rise and Fall Time Mismatch | | | 1.0 | ns |
| $t_{DOETD}$ | DO± End of Transmission | | 200 | 375 | ns |
| $t_{PWODI}$ | DI Pulse Width Accept/Reject Threshold | $\|V_{IN}\| > \|VASQ\|$ (Note 1) | 15 | 45 | ns |
| $t_{PWKDI}$ | DI Pulse Width Maintain/Turn-Off Threshold | $\|V_{IN}\| > \|VASQ\|$ (Note 2) | 136 | 200 | ns |
| $t_{PWOCI}$ | CI Pulse Width Accepth/Reject Threshold | $\|V_{IN}\| > \|VASQ\|$ (Note 3) | 10 | 26 | ns |
| $t_{PWKCI}$ | CI Pulse Width Maintain/Turn-Off Threshold | $\|V_{IN}\| > \|VASQ\|$ (Note 4) | 90 | 160 | ns |
| **Internal MENDEC Clock Timing** | | | | | |
| tx1 | XTAL1 Period | $V_{IN}$ = External Clock | 49.995 | 50.001 | ns |
| tx1H | XTAL1 HIGH Pulse Width | $V_{IN}$ = External Clock | 20 | | ns |
| tx1L | XTAL1 LOW Pulse Width | $V_{IN}$ = External Clock | 20 | | ns |
| tx1R | XTAL1 Rise Time | $V_{IN}$ = External Clock | | 5 | ns |
| tx1F | XTAL1 Fall Time | $V_{IN}$ = External Clock | | 5 | ns |

***Notes:***

1. *DI pulses narrower than $t_{PWODI}$ (min) will be rejected; pulses wider than $t_{PWODI}$ (max) will turn internal DI carrier sense on.*
2. *DI pulses narrower than $t_{PWKDI}$ (min) will maintain internal DI carrier sense on; pulses wider than $t_{PWKDI}$ (max) will turn internal DI carrier sense off.*
3. *CI pulses narrower than $t_{PWOCI}$ (min) will be rejected; pulses wider than $t_{PWOCI}$ (max) will turn internal CI carrier sense on.*
4. *CI pulses narrower than $t_{PWKCI}$ (min) will maintain internal CI carrier sense on; pulses wider than $t_{PWKCI}$ (max) will turn internal CI carrier sense off.*

## AC SWITCHING CHARACTERISTICS

## SCSI Interface

### FastClk Disabled (Control Register Three (0CH) bit 3=0), See Figure 29

| No. | Parameter Symbol | Parameter Description | Test Conditions | Min | Max | Unit |
|---|---|---|---|---|---|---|
| 1 | $t_{PWL}$[1] | Clock Pulse Width Low | | 14.58 | $0.65 \bullet t_{CP}$ | ns |
| 2 | $t_{CP}$ | Clock Period (1 ÷ Clock Frequency) | | 40 | 100 | ns |
| 3 | $t_L$ | Synchronization Latency | | 54.58 | $t_{PWL} + t_{CP}$ | ns |
| 4 | $t_{PWH}$[1] | Clock Pulse Width High | | 14.58 | $0.65 \bullet t_{CP}$ | ns |
| 5 | $t_{RISE}$* | Clock Rise Time | over 2 V p-p | 1 | 4 | V/ns |
| 6 | $t_{FALL}$* | Clock Fall Time | over 2 V p-p | 1 | 4 | V/ns |

***Notes:***

*1. For Synchronous data transmissions, the following conditions must be true:*

*$2t_{CP} + t_{PWL} \geq 97.92$ ns*
*$2t_{CP} + t_{PWH} \geq 97.92$ ns*

*Clock Frequency Range for Fast Clk disabled.*

*= 10 MHz to 25 MHz for Asynchronous Transmission*
*= 12 MHz to 25 MHz for Synchronous Transmission*

** These parameters are not 100% tested, but are evaluated at initial characterization and at any time the design is modified where these parameters may be affected.*

### FastClk Enabled (Control Register Three (0CH) bit 3=1), See Figure 29

| No. | Parameter Symbol | Parameter Description | Test Conditions | Min | Max | Unit |
|---|---|---|---|---|---|---|
| 1 | $t_{PWL}$ | Clock Pulse Width Low | | $0.4 \bullet t_{CP}$ | $0.6 \bullet t_{CP}$ | ns |
| 2 | $t_{CP}$ | Clock Period (1 ÷ Clock Frequency) | | 25 | 50 | ns |
| 3A | $t_L$ | Synchronization Latency | | 54.58 | $2 \bullet t_{CP}$ | ns |
| 4 | $t_{PWH}$ | Clock Pulse Width High | | $0.4 \bullet t_{CP}$ | $0.6 \bullet t_{CP}$ | ns |
| 5 | $t_{RISE}$* | Clock Rise Time | over 2 V p-p | 1 | 4 | V/ns |
| 6 | $t_{FALL}$* | Clock Fall Time | over 2 V p-p | 1 | 4 | V/ns |

***Notes:***

*Clock Frequency Range for Fast Clk enabled.*

*= 20 MHz to 40 MHz for Asynchronous Transmission*
*= 20 MHz to 40 MHz for Synchronous Transmission*

** These parameters are not 100% tested, but are evaluated at initial characterization and at any time the design is modified where these parameters may be affected.*

## AC SWITCHING CHARACTERISTICS (continued)

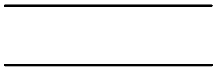
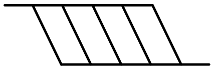
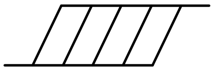
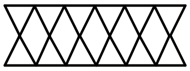
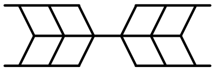
### SCSI Interface

| No. | Parameter Symbol | Parameter Description | Test Conditions | Min | Max | Unit |
|---|---|---|---|---|---|---|
| **Single Ended: Asynchronous Initiator Transmit, See Figure 30** | | | | | | |
| 7 | $t_S$ | Data to $\overline{\text{ACK}}$ ↓ Set Up Time | | 60 | | ns |
| 8 | $t_{PD}$ | $\overline{\text{REQ}}$ ↑ to Data Delay | | 5 | | ns |
| 9 | $t_{PD}$ | $\overline{\text{REQ}}$ ↑ to $\overline{\text{ACK}}$ ↑ Delay | | | 50 | ns |
| 10 | $t_{PD}$ | $\overline{\text{REQ}}$ ↓ to $\overline{\text{ACK}}$ ↓ Delay | | | 50 | ns |
| **Single Ended: Asynchronous Initiator Receive, See Figure 31** | | | | | | |
| 11 | $t_{PD}$ | $\overline{\text{REQ}}$ ↑ to $\overline{\text{ACK}}$ ↑ Delay | | | 50 | ns |
| 12 | $t_{PD}$ | $\overline{\text{REQ}}$ ↓ to $\overline{\text{ACK}}$ ↓ Delay | | | 50 | ns |
| 13 | $t_S$ | Data to $\overline{\text{REQ}}$ ↓ Set Up Time | | 10 | | ns |
| 14 | $t_H$ | $\overline{\text{REQ}}$ ↓ to Data Hold Time | | 18 | | ns |
| **Normal SCSI: (Single Ended) Synchronous Initiator Transmit, See Figure 32** | | | | | | |
| 15 | $t_S$ | Data to $\overline{\text{REQ}}$ or $\overline{\text{ACK}}$ ↓ Set Up Time | | 55* | | ns |
| 16 | $t_{PWL}$ | $\overline{\text{REQ}}$ or $\overline{\text{ACK}}$ Pulse Width Low | | 90* | | ns |
| 17 | $t_{PWH}$ | $\overline{\text{REQ}}$ or $\overline{\text{ACK}}$ Pulse Width High | | 90* | | ns |
| 18 | $t_H$ | $\overline{\text{ACK}}$ or $\overline{\text{REQ}}$ ↓ to Data Hold Time | | 100* | | ns |
| **Fast SCSI: (Single Ended) Synchronous Initiator Transmit, See Figure 32** | | | | | | |
| 15 | $t_S$ | Data to $\overline{\text{REQ}}$ or $\overline{\text{ACK}}$ ↓ Set Up Time | | 25* | | ns |
| 16 | $t_{PWL}$ | $\overline{\text{REQ}}$ or $\overline{\text{ACK}}$ Pulse Width Low | | 30* | | ns |
| 17 | $t_{PWH}$ | $\overline{\text{REQ}}$ or $\overline{\text{ACK}}$ Pulse Width High | | 30* | | ns |
| 18 | $t_H$ | $\overline{\text{ACK}}$ or $\overline{\text{REQ}}$ ↓ to Data Hold Time | | 35* | | ns |
| **Synchronous Initiator Receive, See Figure 33** | | | | | | |
| 19 | $t_{PWL}$ | $\overline{\text{REQ}}$ Pulse Width Low | | 27 | | ns |
| 19 | $t_{PWL}$ | $\overline{\text{ACK}}$ Pulse Width Low | | 20 | | ns |
| 20 | $t_{PWH}$ | $\overline{\text{REQ}}$ Pulse Width High | | 20 | | ns |
| 20 | $t_{PWH}$ | $\overline{\text{ACK}}$ Pulse Width High | | 20 | | ns |
| 21 | $t_S$ | Data to $\overline{\text{REQ}}$ or $\overline{\text{ACK}}$ ↓ Set Up Time | | 10 | | ns |
| 22 | $t_H$ | $\overline{\text{REQ}}$ or $\overline{\text{ACK}}$ ↓ to Data Hold Time | | 15 | | ns |
| **SCSI Bus Lines** | | | | | | |
| | $t_{RISE}$** | Rise Time, 10% to 90% | SCSI Termination + 20 pF | 8 | 20 | ns |
| | $t_{FALL}$** | Fall Time, 10% to 90% | SCSI Termination + 20 pF | 5 | 12 | ns |
| | $dV_H/dt$** | Slew Rate, Low to High | SCSI Termination + 20 pF | 0.15 | 0.50 | V/ns |
| | $dV_L/dt$** | Slew Rate, High to Low | SCSI Termination + 20 pF | 0.25 | 0.80 | V/ns |

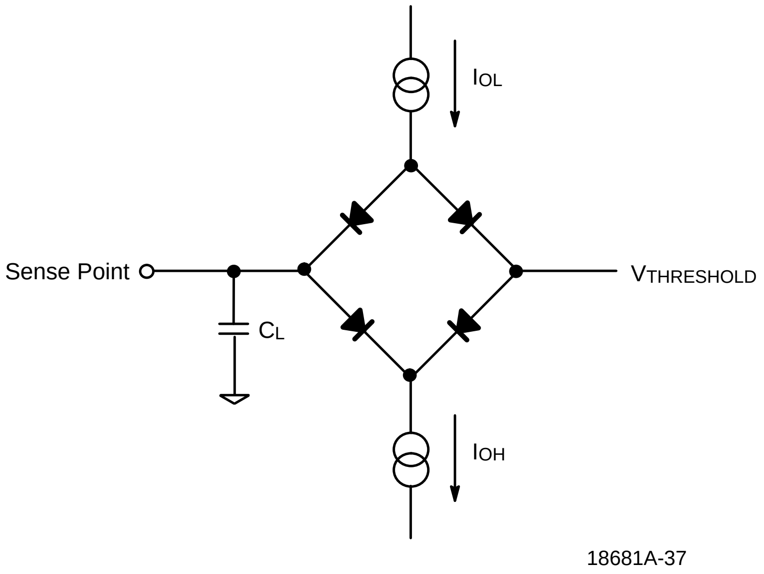
* *The minimum values have a wide range since they depend on the Synchronization latency. The synchronization latency, in turn, depends on the operating frequency of the device.*

***These parameters are not 100% tested, but are evaluated at initial characterization and at any time the design is modified where these parameters may be affected.*

## KEY TO SWITCHING WAVEFORMS

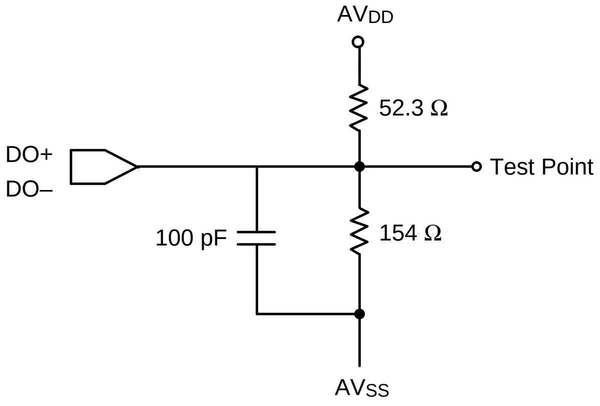
| WAVEFORM | INPUTS | OUTPUTS |
|---|---|---|
|  | Must be Steady | Will be Steady |
|  | May Change from H to L | Will be Changing from H to L |
|  | May Change from L to H | Will be Changing from L to H |
|  | Don’t Care, Any Change Permitted | Changing, State Unknown |
|  | Does Not Apply | Center Line is High-Impedance “Off” State |

## AC SWITCHING TEST CIRCUITS



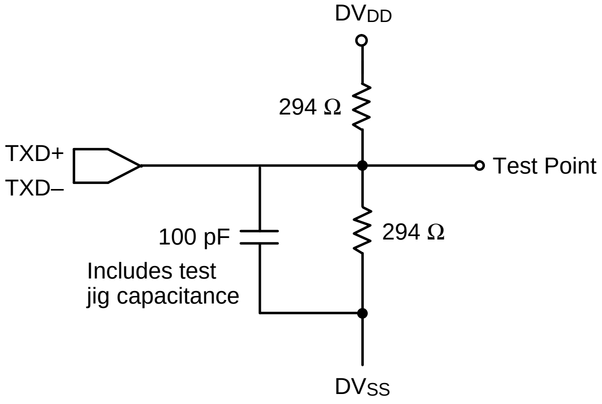
**Normal and Three-State Outputs**

## AC SWITCHING TEST CIRCUITS



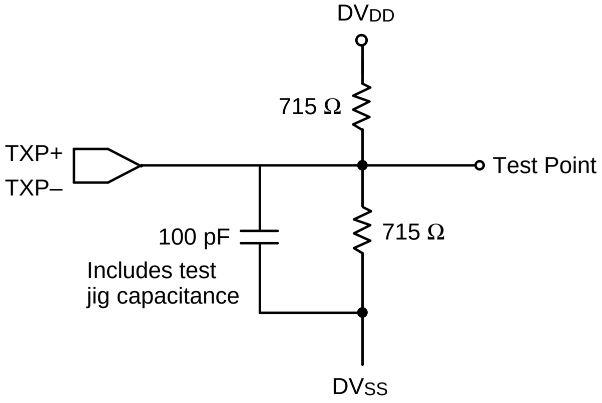
18681A-38

**AUI DO Switching Test Circuit**



18681A-39

**TXD Switching Test Circuit**

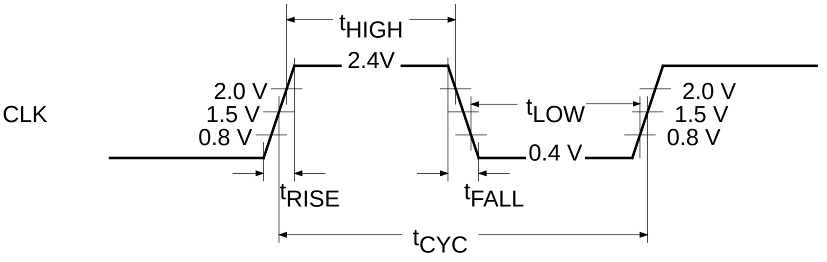


18681A-40

**TXP Outputs Test Circuit**

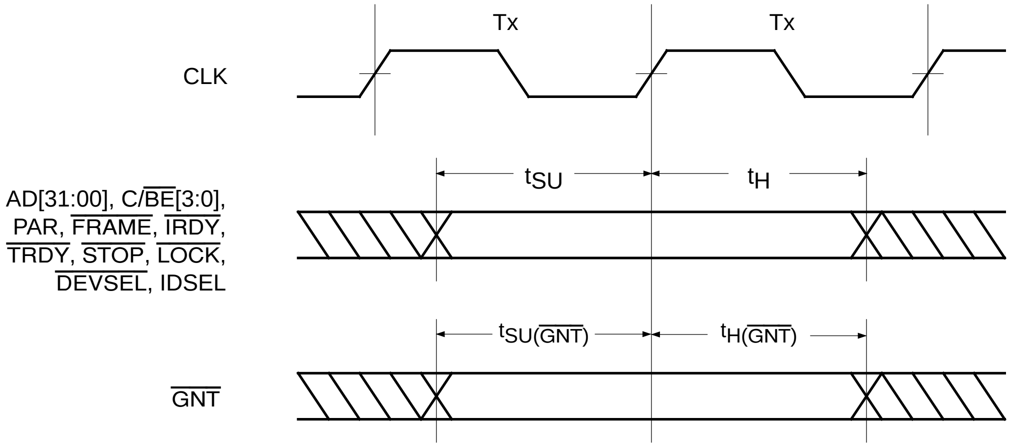
# AC SWITCHING WAVEFORMS

## System Bus Interface



18681A-41

**CLK Waveform**

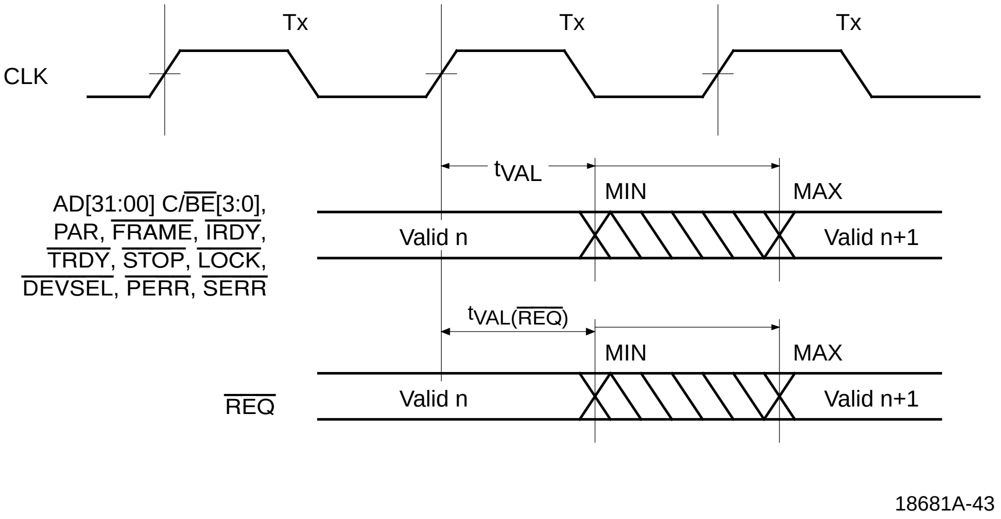

Tx
Tx
CLK
tSU
tH
AD[31:00], C/BE[3:0], PAR, FRAME, IRDY, TRDY, STOP, LOCK, DEVSEL, IDSEL
tSU(GNT)
tH(GNT)
GNT

18681A-42

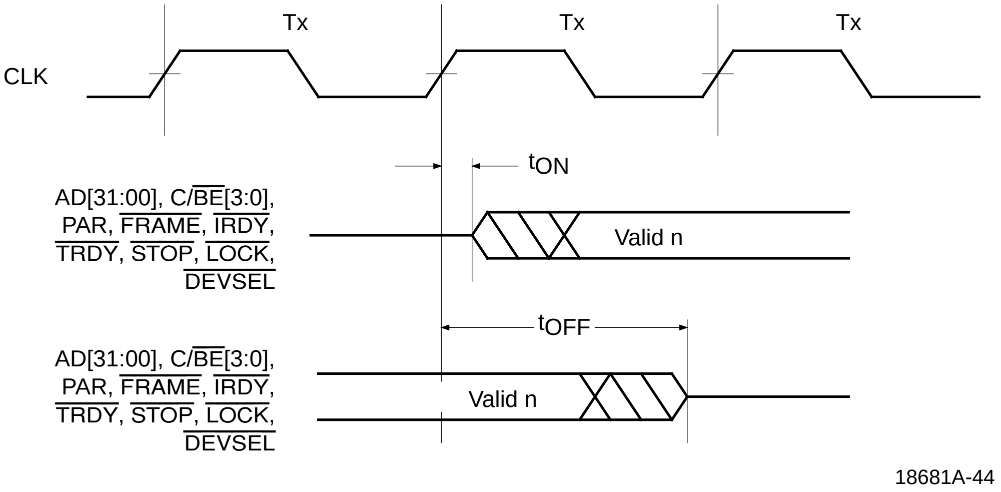
**Input Setup and Hold Timing**

## AC SWITCHING WAVEFORMS

### System Bus Interface



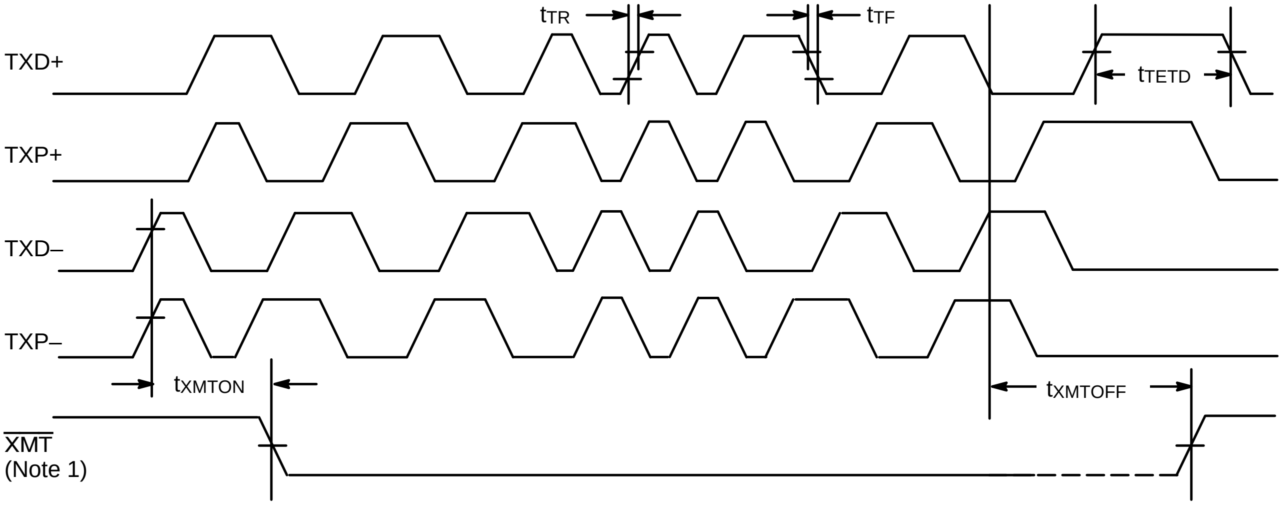
**Output Valid Delay Timing**



**Output Tri-State Delay Timing**

## AC SWITCHING WAVEFORMS

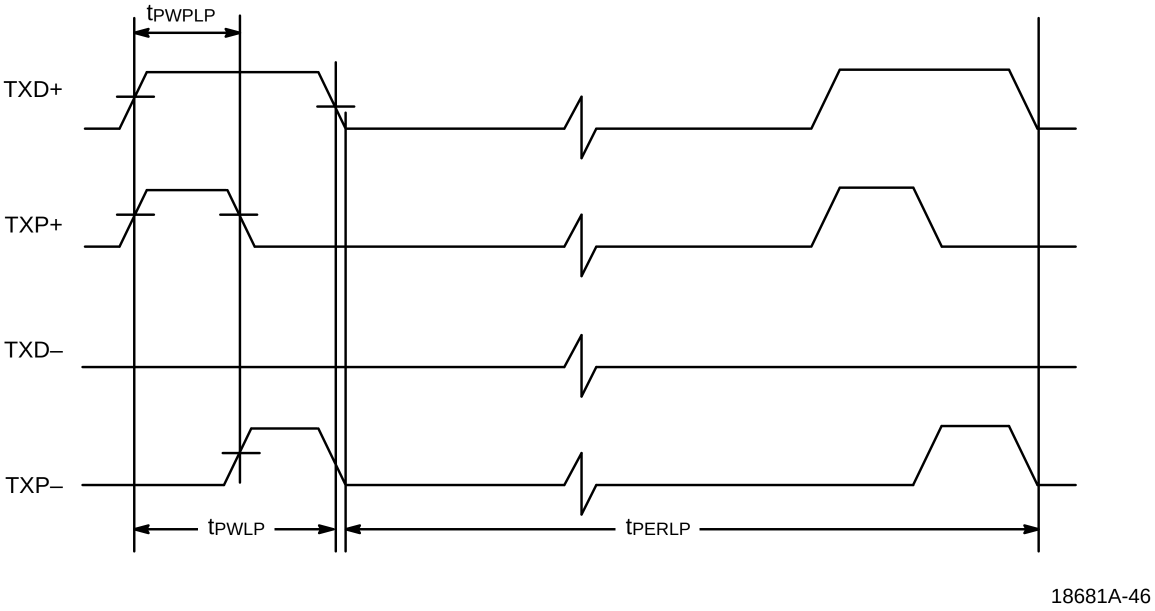
### 10BASE-T Interface



***Note:***

*1. Internal signal and is shown for clarification only.*

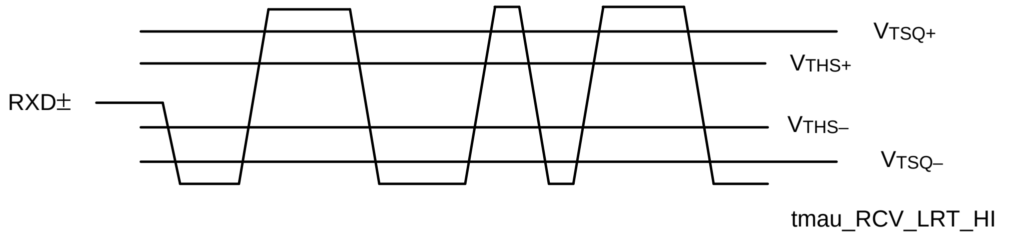
**Transmit Timing**



**Idle Link Test Pulse**

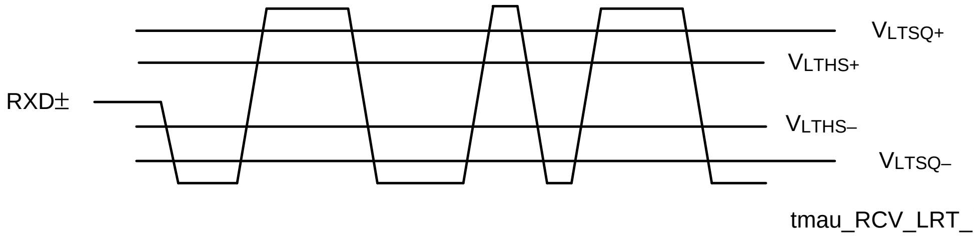
## SWITCHING WAVEFORMS

### 10BASE-T Interface



18681A-47

**Receive Thresholds (LRT=0)**

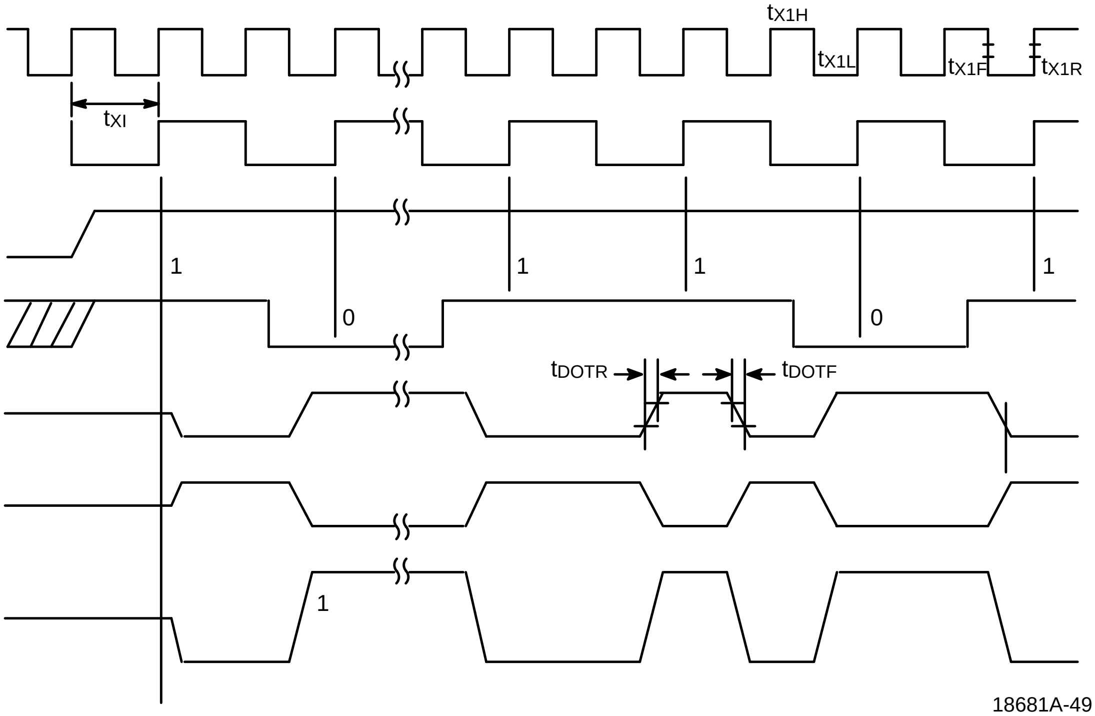


18681A-48

**Receive Thresholds (LRT=1)**

## AC SWITCHING WAVEFORMS

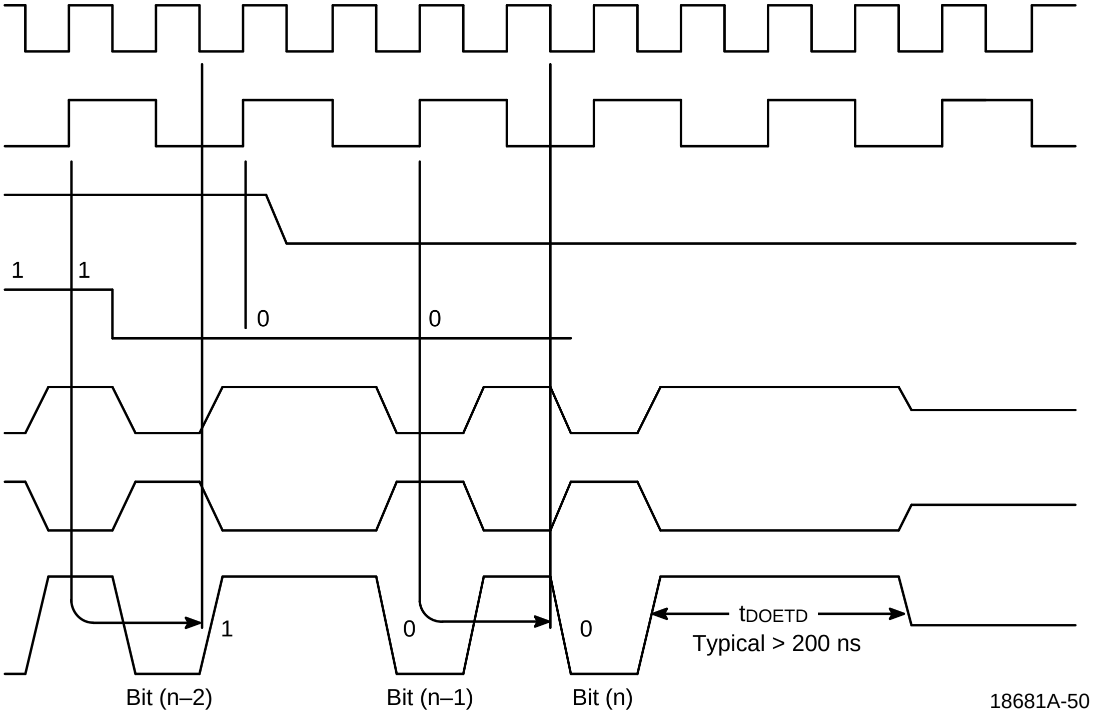
### Attachment Unit Interface



***Note:***

*1. Internal signal and is shown for clarification only.*

**Transmit Timing – Start of Frame**



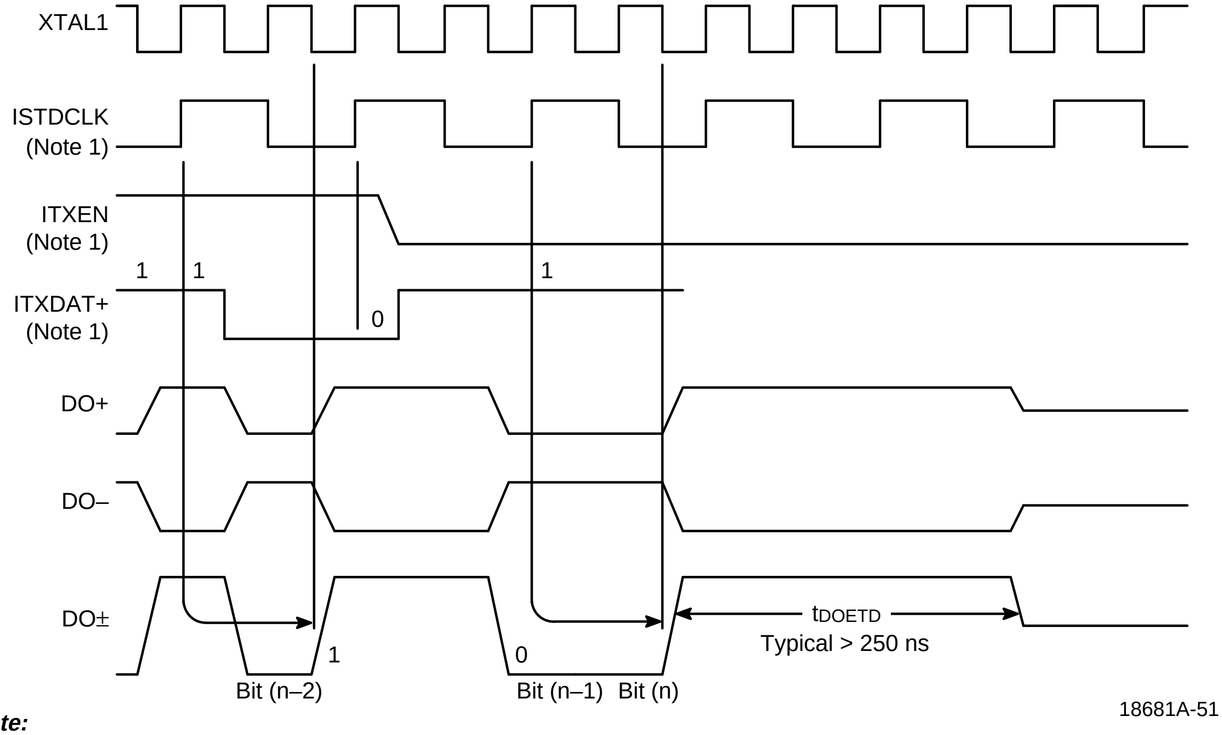
***Note:***

*1. Internal signal and is shown for clarification only.*

**Transmit Timing – End of Frame (Last Bit = 0)**

## SWITCHING WAVEFORMS

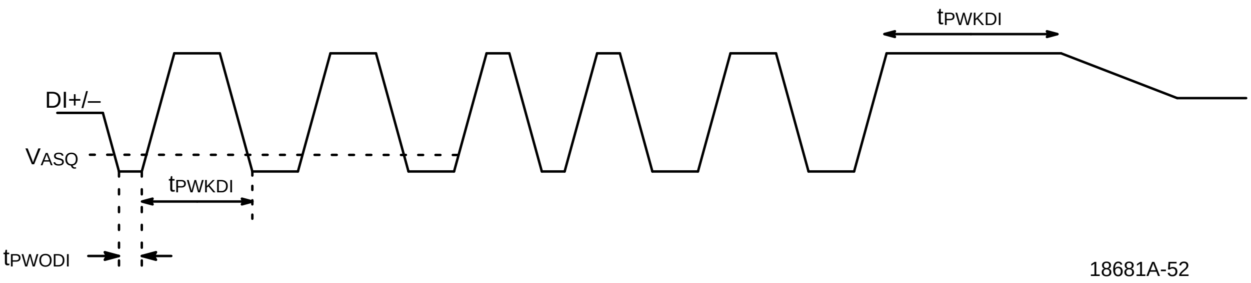
### Attachment Unit Interface



***Note:***

*1. Internal signal and is shown for clarification only.*

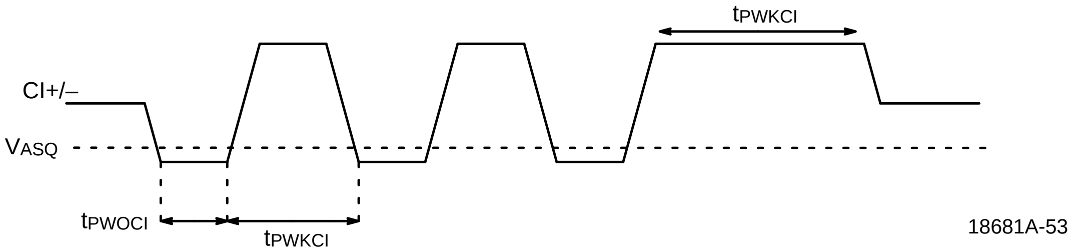
**Transmit Timing – End of Frame (Last Bit = 1)**



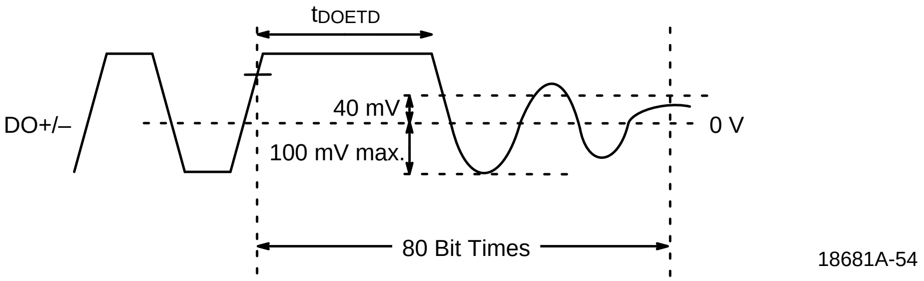
**Receive Timing**

## AC SWITCHING WAVEFORMS

### Attachment Unit Interface



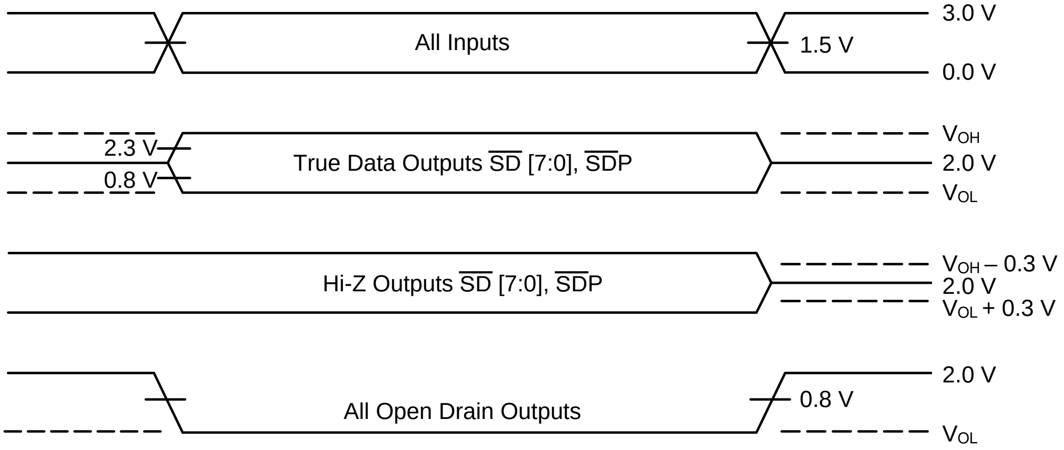
**Collision Timing**



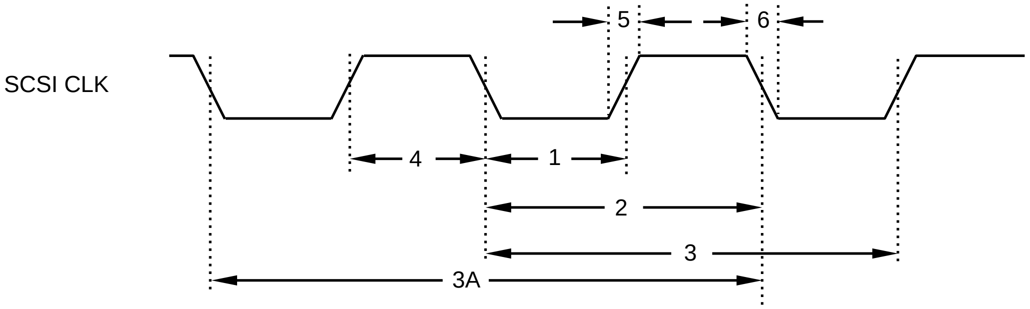
**Port DO ETD Waveform**

## AC SWITCHING TEST WAVEFORMS

### SCSI Interface

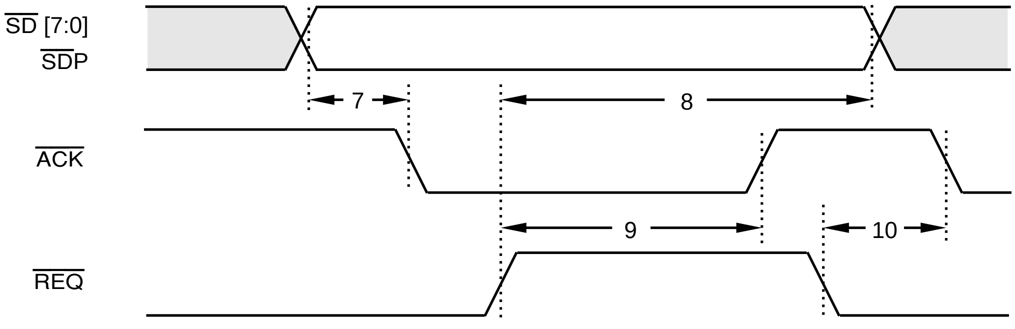

All Inputs
3.0 V
1.5 V
0.0 V
$V_{OH}$
2.3 V
True Data Outputs $\overline{\text{SD}}$ [7:0], $\overline{\text{SDP}}$
2.0 V
0.8 V
$V_{OL}$
Hi-Z Outputs $\overline{\text{SD}}$ [7:0], $\overline{\text{SDP}}$
$V_{OH} – 0.3$ V
2.0 V
$V_{OL} + 0.3$ V
2.0 V
All Open Drain Outputs
0.8 V
$V_{OL}$

18681A-55



18681A-56

**Figure 29. Clock Input**

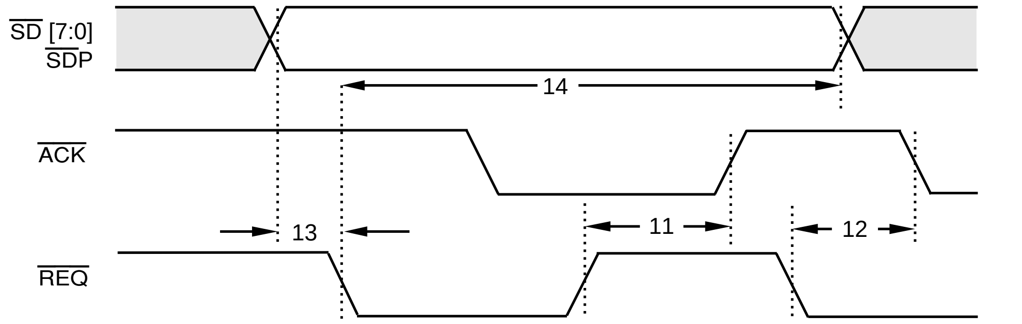


18681A-57

**Figure 30. Asynchronous Initiator Transmit**

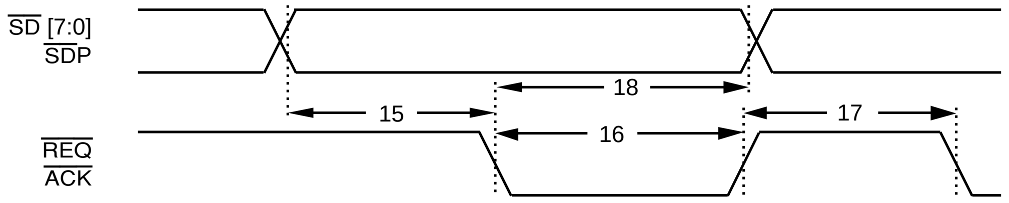
# AC SWITCHING TEST WAVEFORMS

## SCSI Interface



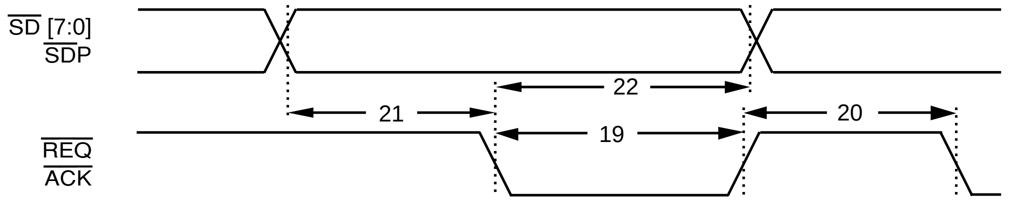
18681A-58

**Figure 31. Asynchronous Initiator Receive**



18681A-59

**Figure 32. Synchronous Initiator Transmit**

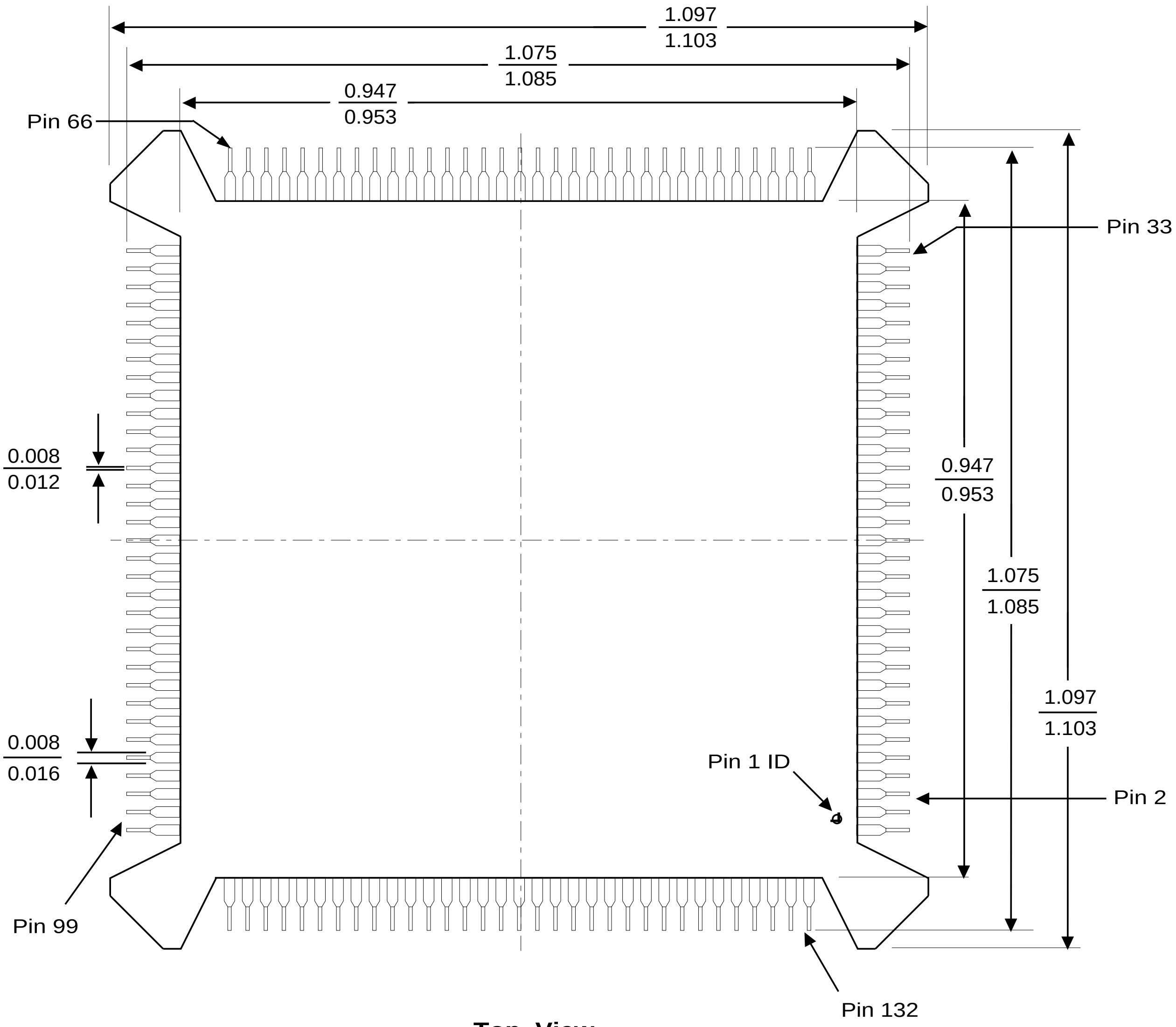


18681A-60

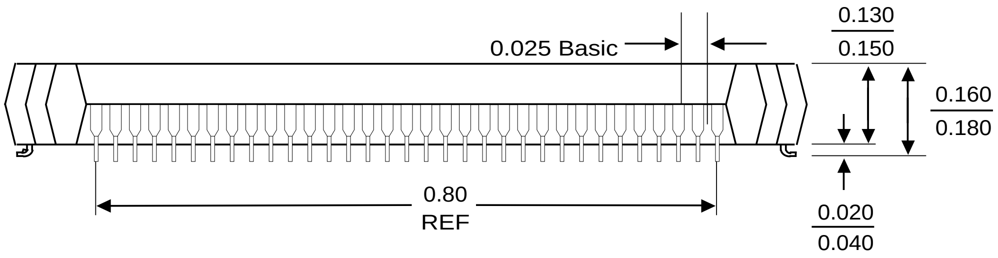
**Figure 33. Synchronous Initiator Receive**

## PHYSICAL DIMENSIONS*

### PQB132
### Plastic Quad Flat Pack Trimmed and Formed (measured in inches)



**Top View**



**Side View**

**For reference only. BSC is an ANSI standard for Basic Space Centering.*

20010A
CL85
08/27/93 MH

**Trademarks**

AMD and the AMD logo are registered trademarks of Advanced Micro Devices, Inc.

Product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

# Register Summary

## Ethernet PCI Configuration Registers

*Note: RO = read only, RW = read/write, U = undefined value*

| Offset | Name | Width in Bit | Access Mode | Default Value |
|---|---|---|---|---|
| 00h | Vendor ID | 16 | RO | 1022h |
| 02h | Device ID | 16 | RO | 2000h |
| 04h | Command | 16 | RW | 0uuu |
| 06h | Status | 16 | RW | uu00 |
| 08h | Revision ID | 8 | RO | 00h |
| 09h | Programming IF | 8 | RO | 00h |
| 0Ah | Sub-Class | 8 | RO | 00h |
| 0Bh | Base-Class | 8 | RO | 02h |
| 0Dh | Latency Timer | 8 | RO | 00h |
| 0Eh | Header Type | 8 | RO | 00h |
| 10h | Base Address | 32 | RW | uuuu uuuu |
| 3Ch | Interrupt Line | 8 | RW | uu |
| 3Dh | Interrupt Pin | 8 | RO | 02h |

## SCSI PCI Configuration Registers

*Note: RO = read only, RW = read/write, U = undefined value*

| Offset | Name | Width in Bit | Access Mode | Default Value |
|---|---|---|---|---|
| 00h | Vendor ID | 16 | RO | 1022h |
| 02h | Device ID | 16 | RO | 2020h |
| 04h | Command | 16 | RW | 0080h |
| 06h | Status | 16 | RW | uuuu |
| 08h | Revision ID | 8 | RO | 00h |
| 09h | Programming IF | 8 | RO | 00h |
| 0Ah | Sub-Class | 8 | RO | 00h |
| 0Bh | Base-Class | 8 | RO | 01h |
| 0Dh | Latency Timer | 8 | RO | 00h |
| 0Eh | Header Type | 8 | RO | 00h |
| 10h | Base Address | 32 | RW | uuuu uuuu |
| 3Ch | Interrupt Line | 8 | RW | uu |
| 3Dh | Interrupt Pin | 8 | RO | 01h |
| 40h | Reserved for Software | 32 | RW | uuuu |
| 44h | Reserved for Software | 32 | RW | uuuu |
| 48h | Reserved for Software | 32 | RW | uuuu |
| 4Ch | Reserved for Software | 32 | RW | uuuu |

## Ethernet Controller

### Control and Status Registers

*Note: u = undefined value, R = Running register, S = Setup register, T = Test register*

| RAP Addr | Symbol | Default Value After H_RESET | Comments | Use |
|---|---|---|---|---|
| 00 | CSR0 | uuuu 0004 | PCnet-SCSI Status Register | R |
| 01 | CSR1 | uuuu uuuu | IADR[15:0]: Base Address of INIT Block Lower | S |
| 02 | CSR2 | uuuu uuuu | IADR[31:16]: Base Address of INIT Block Upper | S |
| 03 | CSR3 | uuuu 0000 | Interrupt Masks and Deferral Control | S |
| 04 | CSR4 | uuuu 0115 | Test and Features Control | R |
| 05 | CSR5 | uuuu 0000 | Reserved | T |
| 06 | CSR6 | uuuu uuuu | RXTX: RX/TX Descriptor Table Lengths | T |
| 07 | CSR7 | uuuu 0000 | Reserved | T |
| 08 | CSR8 | uuuu uuuu | LADR0: Logical Address Filter — LADRF[15:0] | T |
| 09 | CSR9 | uuuu uuuu | LADR1: Logical Address Filter — LADRF[31:16] | T |
| 10 | CSR10 | uuuu uuuu | LADR2: Logical Address Filter — LADRF[47:32] | T |
| 11 | CSR11 | uuuu uuuu | LADR3: Logical Address Filter — LADRF[63:48] | T |
| 12 | CSR12 | uuuu uuuu | PADR0: Physical Address Register — PADR[15:0] | T |
| 13 | CSR13 | uuuu uuuu | PADR1: Physical Address Register — PADR[31:16] | T |
| 14 | CSR14 | uuuu uuuu | PADR2: Physical Address Register — PADR[47:32] | T |
| 15 | CSR15 | see reg. desc. | MODE: Mode Register | S |
| 16 | CSR16 | uuuu uuuu | IADR[15:0]: Alias of CSR1 | T |
| 17 | CSR17 | uuuu uuuu | IADR[31:16]: Alias of CSR2 | T |
| 18 | CSR18 | uuuu uuuu | CRBAL: Current RCV Buffer Address Lower | T |
| 19 | CSR22 | uuuu uuuu | CRBAU: Current RCV Buffer Address Upper | T |
| 20 | CSR20 | uuuu uuuu | CXBAL: Current XMT Buffer Address Lower | T |
| 21 | CSR21 | uuuu uuuu | CXBAU: Current XMT Buffer Address Upper | T |

## Control and Status Registers (continued)

| RAP Addr | Symbol | Default Value After H_RESET | Comments | Use |
|---|---|---|---|---|
| 22 | CSR22 | uuuu uuuu | NRBAL: Next RCV Buffer Address Lower | T |
| 23 | CSR23 | uuuu uuuu | NRBAU: Next RCV Buffer Address Upper | T |
| 24 | CSR24 | uuuu uuuu | BADRL: Base Address of RCV Ring Lower | S |
| 25 | CSR25 | uuuu uuuu | BADRU: Base Address of RCV Ring Upper | S |
| 26 | CSR26 | uuuu uuuu | NRDAL: Next RCV Descriptor Address Lower | T |
| 27 | CSR27 | uuuu uuuu | NRDAU: Next RCV Descriptor Address Upper | T |
| 28 | CSR28 | uuuu uuuu | CRDAL: Current RCV Descriptor Address Lower | T |
| 29 | CSR29 | uuuu uuuu | CRDAU: Current RCV Descriptor Address Upper | T |
| 30 | CSR30 | uuuu uuuu | BADXL: Base Address of XMT Ring Lower | S |
| 31 | CSR31 | uuuu uuuu | BADXU: Base Address of XMT Ring Upper | S |
| 32 | CSR32 | uuuu uuuu | NXDAL: Next XMT Descriptor Address Lower | T |
| 33 | CSR33 | uuuu uuuu | NXDAU: Next XMT Descriptor Address Upper | T |
| 34 | CSR34 | uuuu uuuu | CXDAL: Current XMT Descriptor Address Lower | T |
| 35 | CSR35 | uuuu uuuu | CXDAU: Current XMT Descriptor Address Upper | T |
| 36 | CSR36 | uuuu uuuu | NNRDAL: Next Next Receive Descriptor Address Lower | T |
| 37 | CSR37 | uuuu uuuu | NNRDAU: Next Next Receive Descriptor Address Upper | T |
| 38 | CSR38 | uuuu uuuu | NNXDAL: Next Next Transmit Descriptor Address Lower | T |
| 39 | CSR39 | uuuu uuuu | NNXDAU: Next Next Transmit Descriptor Address Upper | T |
| 40 | CSR40 | uuuu uuuu | CRBC: Current RCV Byte Count | T |
| 41 | CSR41 | uuuu uuuu | CRST: Current RCV Status | T |
| 42 | CSR42 | uuuu uuuu | CXBC: Current XMT Byte Count | T |
| 43 | CSR43 | uuuu uuuu | CXST: Current XMT Status | T |
| 44 | CSR44 | uuuu uuuu | NRBC: Next RCV Byte Count | T |
| 45 | CSR45 | uuuu uuuu | NRST: Next RCV Status | T |
| 46 | CSR46 | uuuu uuuu | POLL: Poll Time Counter | T |
| 47 | CSR47 | uuuu uuuu | POLLINT: Polling Interval | S |
| 48 | CSR48 | uuuu uuuu | Reserved | T |
| 49 | CSR49 | uuuu uuuu | Reserved | T |
| 50 | CSR50 | uuuu uuuu | Reserved | T |
| 51 | CSR51 | uuuu uuuu | Reserved | T |
| 52 | CSR52 | uuuu uuuu | Reserved | T |
| 53 | CSR53 | uuuu uuuu | Reserved | T |
| 54 | CSR54 | uuuu uuuu | Reserved | T |
| 55 | CSR55 | uuuu uuuu | Reserved | T |
| 56 | CSR56 | uuuu uuuu | Reserved | T |
| 57 | CSR57 | uuuu uuuu | Reserved | T |
| 58 | CSR58 | see reg. desc. | SWS: Software Style | S |
| 59 | CSR59 | uuuu 0105 | IR: IR Register | T |
| 60 | CSR60 | uuuu uuuu | PXDAL: Previous XMT Descriptor Address Lower | T |
| 61 | CSR61 | uuuu uuuu | PXDAU: Previous XMT Descriptor Address Upper | T |
| 62 | CSR62 | uuuu uuuu | PXBC: Previous XMT Byte Count | T |
| 63 | CSR63 | uuuu uuuu | PXST: Previous XMT Status | T |
| 64 | CSR64 | uuuu uuuu | NXBA: Next XMT Buffer Address Lower | T |

## Control and Status Registers (continued)

| RAP Addr | Symbol | Default Value After H_RESET | Comments | Use |
|---|---|---|---|---|
| 65 | CSR65 | uuuu uuuu | NXBAU: Next XMT Buffer Address Upper | T |
| 66 | CSR66 | uuuu uuuu | NXBC: Next XMT Byte Count | T |
| 67 | CSR67 | uuuu uuuu | NXST: Next XMT Status | T |
| 68 | CSR68 | uuuu uuuu | Reserved | T |
| 69 | CSR69 | uuuu uuuu | Reserved | T |
| 70 | CSR70 | uuuu uuuu | Reserved | T |
| 71 | CSR71 | uuuu uuuu | Reserved | T |
| 72 | CSR72 | uuuu uuuu | RCVRC: RCV Ring Counter | T |
| 73 | CSR73 | uuuu uuuu | Reserved | T |
| 74 | CSR74 | uuuu uuuu | XMTRC: XMT Ring Counter | T |
| 75 | CSR75 | uuuu uuuu | Reserved | T |
| 76 | CSR76 | uuuu uuuu | RCVRL: RCV Ring Length | S |
| 77 | CSR77 | uuuu uuuu | Reserved | T |
| 78 | CSR78 | uuuu uuuu | XMTRL: XMT Ring Length | S |
| 79 | CSR79 | uuuu uuuu | Reserved | T |
| 80 | CSR80 | uuuu E810 | DMATCFW: DMA Transfer Counter and FIFO Threshold | S |
| 81 | CSR81 | uuuu uuuu | Reserved | T |
| 82 | CSR82 | uuuu 0000 | DMABAT: Bus Activity Timer | S |
| 83 | CSR83 | uuuu uuuu | Reserved | T |
| 84 | CSR84 | uuuu uuuu | DMABAL: DMA Address Register Lower | T |
| 85 | CSR85 | uuuu uuuu | DMABAU: DMA Address Register Upper | T |
| 86 | CSR86 | uuuu uuuu | DMABC: Buffer Byte Counter | T |
| 87 | CSR87 | uuuu uuuu | Reserved | T |
| 88 | CSR88 | 0242 0003 | Chip ID Register Lower | T |
| 89 | CSR89 | uuuu 0242 | Chip ID Register Upper | T |
| 90 | CSR90 | uuuu uuuu | RAEO Register | S |
| 91 | CSR91 | uuuu uuuu | Reserved | T |
| 92 | CSR92 | uuuu uuuu | RCON: Ring Length Conversion | T |
| 93 | CSR93 | uuuu uuuu | Reserved | T |
| 94 | CSR94 | uuuu 0000 | XMTTDR: Transmit Time Domain Reflectometry Count | T |
| 95 | CSR95 | uuuu uuuu | Reserved | T |
| 96 | CSR96 | uuuu uuuu | Reserved | T |
| 97 | CSR97 | uuuu uuuu | Reserved | T |
| 98 | CSR98 | uuuu uuuu | Reserved | T |
| 99 | CSR99 | uuuu uuuu | Reserved | T |
| 100 | CSR100 | uuuu 0200 | MERRTO: Bus Time-Out | S |
| 101 | CSR101 | uuuu uuuu | Reserved | T |
| 102 | CSR102 | uuuu uuuu | Reserved | T |
| 103 | CSR103 | uuuu 0105 | Reserved | T |
| 104 | CSR104 | uuuu uuuu | Reserved | T |
| 105 | CSR105 | uuuu uuuu | Reserved | T |
| 106 | CSR106 | uuuu uuuu | Reserved | T |
| 107 | CSR107 | uuuu uuuu | Reserved | T |

### Control and Status Registers (continued)

| RAP Addr | Symbol | Default Value After H_RESET | Comments | Use |
|---|---|---|---|---|
| 108 | CSR108 | uuuu uuuu | Reserved | T |
| 109 | CSR109 | uuuu uuuu | Reserved | T |
| 110 | CSR110 | uuuu uuuu | Reserved | T |
| 111 | CSR111 | uuuu uuuu | Reserved | T |
| 112 | CSR112 | uuuu 0000 | MFC: Missed Frame Count | R |
| 113 | CSR113 | uuuu uuuu | Reserved | T |
| 114 | CSR114 | uuuu 0000 | RCC: Receive Collision Count | R |
| 115 | CSR115 | uuuu uuuu | Reserved | T |
| 116 | CSR116 | uuuu uuuu | Reserved | T |
| 117 | CSR117 | uuuu uuuu | Reserved | T |
| 118 | CSR118 | uuuu uuuu | Reserved | T |
| 119 | CSR119 | uuuu uuuu | Reserved | T |
| 120 | CSR120 | uuuu uuuu | Reserved | T |
| 121 | CSR121 | uuuu uuuu | Reserved | T |
| 122 | CSR122 | see reg. desc. | Receive Frame Alignment Control | S |
| 123 | CSR123 | uuuu uuuu | Reserved | T |
| 124 | CSR124 | see reg. desc. | Test Register 1 | T |
| 125 | CSR125 | uuuu uuuu | Reserved | T |
| 126 | CSR126 | uuuu uuuu | Reserved | T |
| 127 | CSR127 | uuuu uuuu | Reserved | T |

### BCR—Bus Configuration Registers

Writes to those registers marked as “Reserved” will have no effect. Reads from these locations will produce undefined values.

| BCR | MNEMONIC | Default | Description | Programmability | |
|---|---|---|---|---|---|
| | | | | User | EEPROM |
| 0 | MSRDA | 0005h | Reserved | No | No |
| 1 | MSWRA | 0005h | Reserved | No | No |
| 2 | MC | N/A* | Miscellaneous Configuration | Yes | Yes |
| 3 | Reserved | N/A | Reserved | No | No |
| 4 | LNKST | 00C0h | Link Status (Default) | Yes | No |
| 5 | LED1 | 0084h | Receive Status (Default) | Yes | No |
| 6 | LED2 | 0088h | Reserved | Yes | No |
| 7 | LED3 | 0090h | Transmit Status (Default) | Yes | No |
| 8–15 | Reserved | N/A | Reserved | No | No |
| 16 | IOBASEL | N/A* | Reserved | Yes | Yes |
| 17 | IOBASEU | N/A* | Reserved | Yes | Yes |
| 18 | BSBC | 2101h | Burst Size and Bus Control | Yes | Yes |
| 19 | EECAS | 0002h | EEPROM Control and Status | Yes | No |
| 20 | SWS | 0000h | Software Style | Yes | No |
| 21 | INTCON | N/A* | Reserved | Yes | Yes |

* *Registers marked with an “*” have no default value, since they are not observable without first being programmed through the EEPROM read operation. Therefore, the only observable values for these registers are those that have been programmed and a default value is not applicable.*

# SCSI Controller

## SCSI Register Map

| Register Acronym | Address (Hex.) | Register Description | Type |
|---|---|---|---|
| CTCREG | (B)+00 | Current Transfer Count Register Low | Read |
| STCREG | (B)+00 | Start Transfer Count Register Low | Write |
| CTCREG | (B)+04 | Current Transfer Count Register Middle | Read |
| STCREG | (B)+04 | Start Transfer Count Register Middle | Write |
| FFREG | (B)+08 | SCSI FIFO Register | Read/Write |
| CMDREG | (B)+0C | SCSI Command Register | Read/Write |
| STATREG | (B)+10 | SCSI Status Register | Read |
| SDIDREG | (B)+10 | SCSI Destination ID Register | Write |
| INSTREG | (B)+14 | Interrupt Status Register | Read |
| STIMREG | (B)+14 | SCSI Timeout Register | Write |
| ISREG | (B)+18 | Internal State Register | Read |
| STPREG | (B)+18 | Synchronous Transfer Period Register | Write |
| CFISREG | (B)+1C | Current FIFO/Internal State Register | Read |
| SOFREG1 | (B)+1C | Synchronous Offset Register | Write |
| CNTLREG1 | (B)+20 | Control Register One | Read/Write |
| CLKFREG | (B)+24 | Clock Factor Register | Write |
| RES | (B)+28 | Reserved | Write |
| CNTLREG2 | (B)+2C | Control Register Two | Read/Write |
| CNTLREG3 | (B)+30 | Control Register Three | Read/Write |
| CNTLREG4 | (B)+34 | Control Register Four | Read/Write |
| CTCREG | (B)+38 | Current Transfer Count Register High/Part-Unique ID Code | Read |
| STCREG | (B)+38 | Start Current Transfer Count Register High | Write |
| RES | (B)+3C | Reserved | Write |

## DMA Register Map

| Register Acronym | Address (Hex.) | Register Description | Type |
|---|---|---|---|
| CMD | (B)+40 | Command | R/W |
| STC | (B)+44 | Starting Transfer Count | R/W |
| SPA | (B)+48 | Starting Physical Address | R/W |
| WBC | (B)+4C | Working Byte Counter | R |
| WAC | (B)+50 | Working Address Counter | R |
| STATUS | (B)+54 | Status Register | R |
| SMDLA | (B)+58 | Starting Memory Descriptor List (MDL) Address | R/W |
| WMAC | (B)+5C | Working MDL Counter | R |

# PCnet-SCSI Compatible Media Interface Modules

## PCnet-SCSI COMPATIBLE 10BASE-T Filters and Transformers

| Manufacturer | Part # | Package | Description |
|---|---|---|---|
| Bel Fuse | A556-2006-DE | 16 pin 0.3” DIL | Transmit and receive filters and transformers. |
| Bel Fuse | 0556-2006-00 | 14-pin SIP | Transmit and receive filters and transformers. |
| Bel Fuse | 0556-2006-01 | 14-pin SIP | Transmit and receive filters, transformers and common mode chokes. |
| Bel Fuse | 0556-6392-00 | 16-pin 0.5” DIL | Transmit and receive filters, transformers and common mode chokes. |
| Halo Electronics | FD02-101G | 16-pin 0.3” DIL | Transmit and receive filters and transformers. |
| Halo Electronics | FD12-101G | 16-pin 0.3” DIL | Transmit and receive filters and transformers, transmit common mode choke. |
| Halo Electronics | FD22-101G | 16-pin 0.3” DIL | Transmit and receive filters, transformers and common mode chokes. |
| PCA Electronics | EPA1990A | 16-Pin 0.3” DIL | Transmit and receive filters and transformers. |
| PCA Electronics | EPA2013D | 16-Pin 0.3” DIL | Transmit and receive filters and transformers, transmit common mode choke. |
| Pulse Engineering | PE-65434 | 10-pin SIP | Transmit and receive filters, transformers, and common mode choke. |
| Pulse Engineering | PE-65445 | 16-pin 0.3” DIL | Transmit and receive filters and transformers (for SMT use PE-65446) |
| Pulse Engineering | PE65467 | 16-Pin 0.3” DIL | Transmit and receive filters, transformers, common mode chokes, and AMD specified resistors. |
| Pulse Engineering | PE-65424 | 16-Pin 0.3” DIL | Transmit and receive filters, transformers, and common mode chokes. |
| TDK | TLA 470 | 14-pin SIP | Transmit and receive filters and transformers. |
| TDK | HIM3000 | 24-pin 0.6” DIL | Transmit and receive filters, transformers and common mode chokes. |
| Valor Electronics | PT3877 | 16-pin 0.3” DIL | Transmit and receive filters and transformers. |
| Valor Electronics | PT3983 | 8-pin 0.3” DIL | Transmit and receive common mode chokes. |
| Valor Electronics | FL1012 | 16-pin 0.3” DIL | Transmit and receive filters and transformers, transmit common mode choke. |

## PCnet-SCSI COMPATIBLE AUI

### Isolation Transformers

| Manufacturer | Part # | Package | Description |
|---|---|---|---|
| Bel Fuse | A553-0506-AB | 16-Pin 0.3” DIL | 50 µH |
| Halo Electronics | TD01-0756K | 16-Pin 0.3” DIL | 75 µH |
| Halo Electronics | TG01-0756W | 16-Pin 0.3” SMD | 75 µH |
| PCA Electronics | EP9531-4 | 16-Pin 0.3” DIL | 50 µH |
| Pulse Engineering | PE64106 | 16-Pin 0.3” DIL | 50 µH |
| TDK | TLA 100-3E | 16-Pin 0.3” DIL | 100 µH |
| Valor Electronics | LT6031 | 16-Pin 0.3” DIL | 50 µH |

## MANUFACTURER CONTACT INFORMATION

Contact the following companies for further information on their products:

| | | |
|---|---|---|
| Bel Fuse | Phone: | (317) 831-4226 |
| | FAX: | (317) 831-4547 |
| Halo Electronics | Phone: | (415) 989-7313 |
| | FAX: | (415) 367-7158 |
| PCA Electronics | Phone: | (408) 954-0400 |
| | FAX: | (408) 954-1440 |
| Pulse Engineering | Phone: | (619) 674-8218 |
| | FAX: | (619) 675-8262 |
| TDK | Phone: | (213) 530-9397 |
| | FAX: | (213) 530-8127 |
| Valor Electronics | Phone: | (619) 458-1471 |
| | FAX: | (619) 458-0875 |

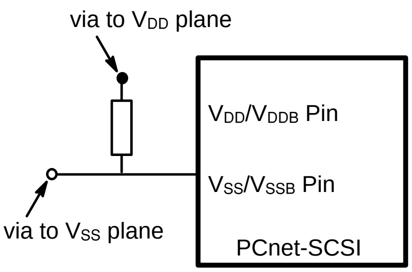
# Recommendation for Power and Ground Decoupling

The PCnet-SCSI controller is an integrated, combination Ethernet and Fast SCSI controller, which contains both digital and analog circuitry. The analog circuitry contains a high speed Phase-Locked Loop (PLL) and Voltage Controlled Oscillator (VCO). Because of the mixed signal characteristics of this chip, some extra precautions must be taken into account when designing with this device.

## Digital Decoupling

The PCnet-SCSI controller separates the digital power supply pins into two groups. The $V_{SSB}$ and $V_{DDB}$ pins supply power to the I/O buffers that connect to the PCI bus. The $V_{SS}$ and $V_{DD}$ pins supply power to all internal circuitry. AMD recommends that at least one low frequency bulk decoupling capacitor be used for each group of digital power pins. 22 µF capacitors have worked well for this. In addition, a total of four or five 0.1 µF capacitors have proven sufficient around $V_{SSB}$ and $V_{DDB}$ pins that supply the drives of the PCI bus output pins. An additional two to the three 0.1 µF capacitors shoud be used around the $V_{SS}$ and $V_{DD}$ pins.

The CMOS technology used in fabricating the PCnet-SCSI controller employs an n-type substrate. In this technology, all $V_{DD}$ and $V_{DDB}$ pins are electrically connected to each other internally. Hence, in a multi-layer board, when decoupling between $V_{DD}/V_{DDB}$ and critical $V_{SS}/V_{SSB}$ pins, the specific $V_{DD}/V_{DDB}$ pin that you connect to is not critical. In fact, the $V_{DD}/V_{DDB}$ connection of the decoupling capacitor can be made directly to the power plane, near the closest $V_{DD}/V_{DDB}$ pin to the $V_{SS}/V_{SSB}$ pin of interest. However, we recommend that the $V_{SS}/V_{SSB}$ connection of the decoupling capacitor be made directly to the $V_{SS}/V_{SSB}$ pin of interest as shown.



18681A-61

AMD recommends that at least one low-frequency bulk decoupling capacitor be used in the area of the PCnet-SCSI controller. 22 µF capacitors have worked well for this. In addition, a total of four or five 0.1 µF capacitors have proven sufficient around the $V_{SSB}$ and $V_{DDB}$ pins that supply the drivers of the PCI bus output pins.

## Analog Decoupling

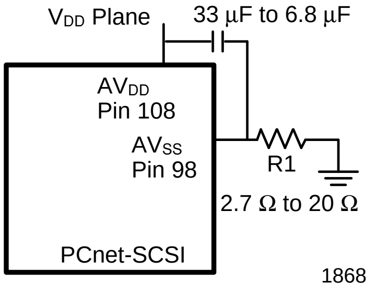
The most critical pins are the analog supply and ground pins. All of the analog supply and ground pins are located in one corner of the device. Specific requirements of the analog supply pins are listed below.

### $AV_{SS1}$ (Pin 100) and $AV_{DD3}$ (Pin 96)

These pins provide the power and ground for the Twisted Pair and AUI drivers. Hence, they are very noisy. A dedicated 0.1 µF capacitor between these pins is recommended.

### $AV_{SS2}$ (Pin 98) and $AV_{DD2}$ (Pin 108)

These pins are the most critical pins on the PCnet-SCSI controller because they provide the power and ground for the PLL portion of the chip. The VCO portion of the PLL is sensitive to noise in the 60 kHz –200 kHz. range. To prevent noise in this frequency range from disrupting the VCO, AMD strongly recommends that the low-pass filter shown below be implemented on these pins. Tests using this filter have shown significantly increased noise immunity and reduced Bit Error Rate (BER) statistics in designs using the PCnet-SCSI controller.



18681A-62

To determine the value for the resistor and capacitor, the formula is:

$$R * C \geq 88$$

Where R is in ohms and C is in microfarads. Some possible combinations are given below. To minimize the voltage drop across the resistor, the R value should not be more than 20 Ω.

| R | C |
|---|---|
| 2.7 Ω | 33 µF |
| 4.3 Ω | 22 µF |
| 6.8 Ω | 15 µF |
| 10 Ω | 10 µF |
| 20 Ω | 6.8 µF |

### $AV_{DD1}$ (Pin 103) and $AV_{DD4}$ (Pin 91)

These pins provide power for the AUI and twisted-pair receive circuitry. No specific decoupling has been necessary on these pins.

# Alternative Method for Initialization of Ethernet Controller

The Ethernet portion of the PCnet-SCSI controller may be initialized by performing I/O writes only. That is, data can be written directly to the appropriate control and status registers (CSR) instead of reading from the initialization Block in memory. The registers that must be written are shown in the table below. These register writes are followed by writing the START bit in CSR0.

| Control and Status Register | Comment |
|---|---|
| CSR8 | LADRF[15:0] |
| CSR9 | LADRF[31:16] |
| CSR10 | LADRF[47:32] |
| CSR11 | LADRF[63:48] |
| CSR12 | PADR[15:0] |
| CSR13 | PADR[31:16] |
| CSR14 | PADR[47:32] |
| CSR15 | Mode |
| CSR24-25 | BADR |
| CSR30-31 | BADX |
| CSR47 | POLLINT |
| CSR76 | RCVRL |
| CSR78 | XMTRL |

***Note****:*

*The INIT bit must not be set or the initialization block will be accessed instead.*

# SCSI System Considerations

## INTRODUCTION

This appendix covers motherboard design considerations which use the Am79C974. It discusses SCSI component placement, signal routing, PCI interface recommendations, noise considerations and termination schemes.

## SIGNAL ROUTING AND SCSI PLACEMENT

The main components for board design include the Am79C974 (whose maximum trace length from the PCI speedway is 1.5″), SCSI connectors (either one or two depending internal/external support), a 40 MHz crystal oscillator and regulated terminators (on-board) which can be up to .1 m (3.937 in) away from the Am79C974.

There are many ways to route SCSI bus traces on a host adapter board or motherboard. Ideally, traces from the internal SCSI bus connector, from the Am79C974 and from the external SCSI bus connector should all connect in series. Care should be taken not to have any stubs in the SCSI bus. (Stubs are any extensions off of the main bus. The maximum SCSI bus stub length allowed is .1 m). This routing scheme helps maintain signal integrity by reducing the possibility of signal reflections and other undesirable effects. Auto-routing programs used for board layout may not follow this scheme, and may create “non-ideal” environments by routing internal and external on-board connectors first instead of routing both sets of traces to the Am79C974. When peripherals are added either internally or externally, a “three-pronged” SCSI bus will be created instead of a linear one. If this or any other stub problem occurs, changes should be made manually to follow the ideal scheme. Refer to Figures E-1 and E-2.

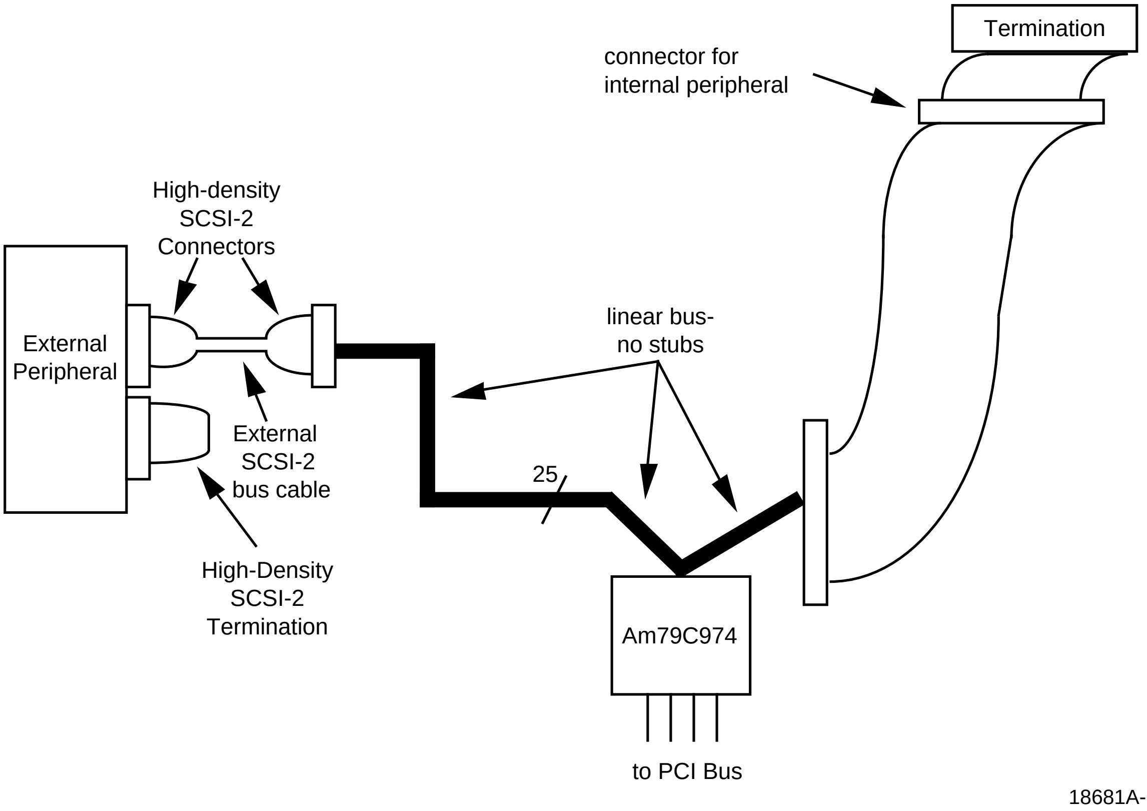

Termination
connector for internal peripheral
High-density SCSI-2 Connectors
External Peripheral
linear bus- no stubs
External SCSI-2 bus cable
25
High-Density SCSI-2 Termination
Am79C974
to PCI Bus
18681A-63

**Figure E-1. Ideal Routing Scheme**

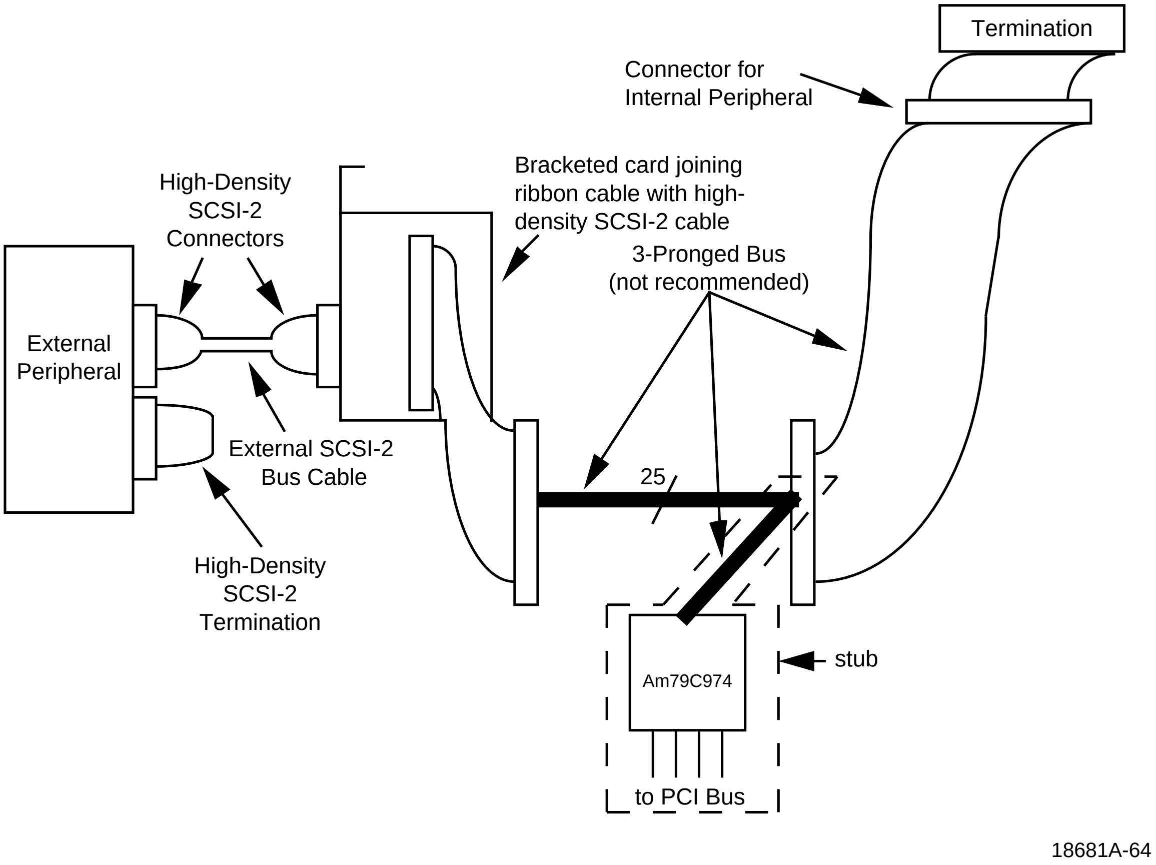


**Figure E-2. A Poor Routing Scheme**

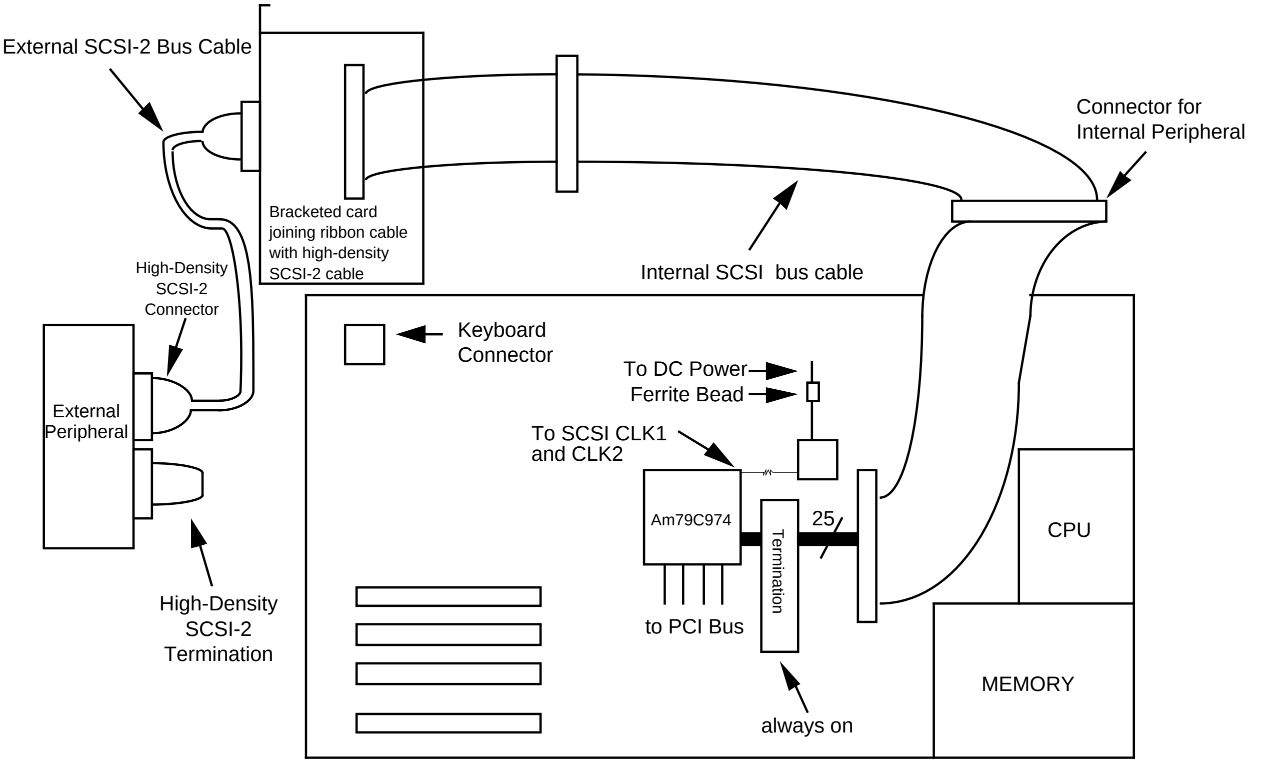
## The Motherboard

The following two layouts may be used as a guideline for the design of motherboards which incorporate the Am79C974. For both layouts, the SCSI connectors should be placed as far away from the PCI Speedway as possible. Each layout refers to termination considerations which are described in further detail in the Termination Considerations section.

### Layout #1

This approach avoids the cost of placing an external connector on the motherboard. In this configuration, the Am79C974 is always at one end of the SCSI bus, therefore, the regulated terminators remain active. This eliminates the problem of switching the regulated terminators on or off to accommodate peripheral configurations. This approach also preserves the ideal linear routing scheme. The following lists the requirements for implementation, while Figure E-3 illustrates this approach.

- One connector on the motherboard connected to one end of the internal bus ribbon cable and its components.
- The other end of the internal bus ribbon cable connected to one end of the external bus high density cable and its peripherals via a bracketed add-on card. The internal bus may also be crimped to a SCSI connector mounted on a bracket.
- On board regulated terminators a maximum of 0.1 m (3.937 in) from the Am79C974.
- External terminators connected to the end of the external bus.


External SCSI-2 Bus Cable
Bracketed card joining ribbon cable with high-density SCSI-2 cable
Connector for Internal Peripheral
Internal SCSI bus cable
High-Density SCSI-2 Connector
External Peripheral
Keyboard Connector
To DC Power
Ferrite Bead
To SCSI CLK1 and CLK2
Am79C974
Termination
25
CPU
to PCI Bus
MEMORY
High-Density SCSI-2 Termination
always on

18681A-65

**Figure E-3. Motherboard Layout — Approach #1**

### Layout #2

This approach uses a “pizza-box” type of motherboard, which has been incorporated into PC systems and workstations. This design reduces the system’s height so that its casing resembles a pizza box. This is partly the result of a riser card that enables cards to rest on their side instead of upright. This approach requires the following and is illustrated in Figure E-4:

- An external connector mounted on the motherboard with routings that connect to the Am79C974.
- The Am79C974 must be as close as possible to this external connector since this part of the SCSI bus consists of motherboard routings (not just ribbon cable).
- An internal connector on the motherboard to accommodate internal drives.
- On-board regulated terminators for SCSI drives not mounted within the system. However, should the user decide to connect a drive internally, terminators are not needed.
- If on-board regulated terminators are used, they should be placed within .1 m (3.937 in) from the Am79C974.

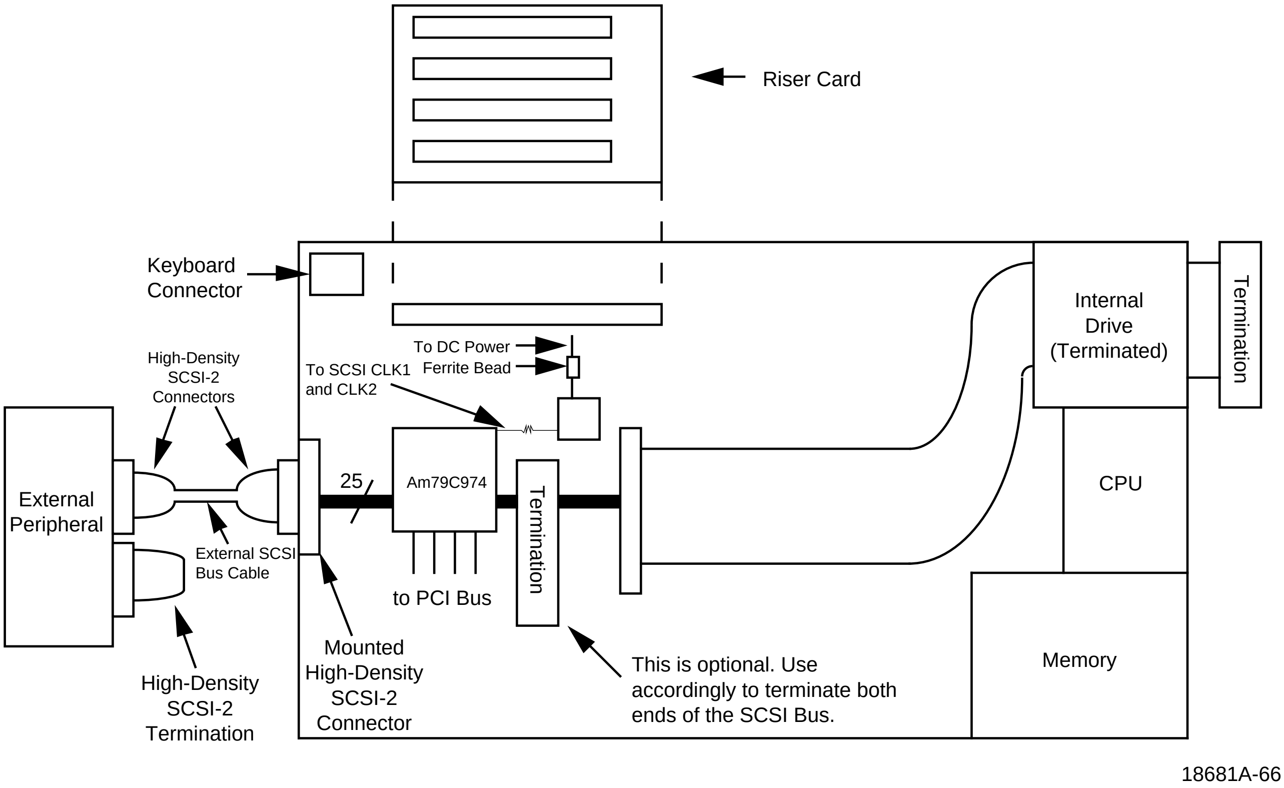

Riser Card
Keyboard Connector
High-Density SCSI-2 Connectors
To DC Power
Ferrite Bead
To SCSI CLK1 and CLK2
External Peripheral
25
Am79C974
Termination
Internal Drive (Terminated)
Termination
CPU
External SCSI Bus Cable
to PCI Bus
High-Density SCSI-2 Termination
Mounted High-Density SCSI-2 Connector
This is optional. Use accordingly to terminate both ends of the SCSI Bus.
Memory
18681A-66

**Figure E-4. Motherboard Layout — Approach #2**

Motherboard designs which place an internal and external connector on either side of the Am79C974 are discouraged since:

- Two SCSI connectors are required on the motherboard—one more than what the other two approaches call for.
- When the number of internal and external bus components increase, the on-board terminators would have to be turned off either through software or hardware, which may be undesirable to a board designer.
- The possibility for creating stubs exists if the connectors and Am79C974 are not routed correctly. See Figure E-1 for the ideal routing scheme.

## NOISE CONSIDERATIONS

Some areas of a PCI motherboard or host adapter design (which include the Am79C974) are more susceptible to noise. They are:

- the 40 MHz crystal oscillator
- the SCSI cables
- the DC power planes
- pins on the ICs, specifically the Am79C974.

### Electromagnetic Interference (EMI)

There are several ways to reduce the amount of noise present in these areas:

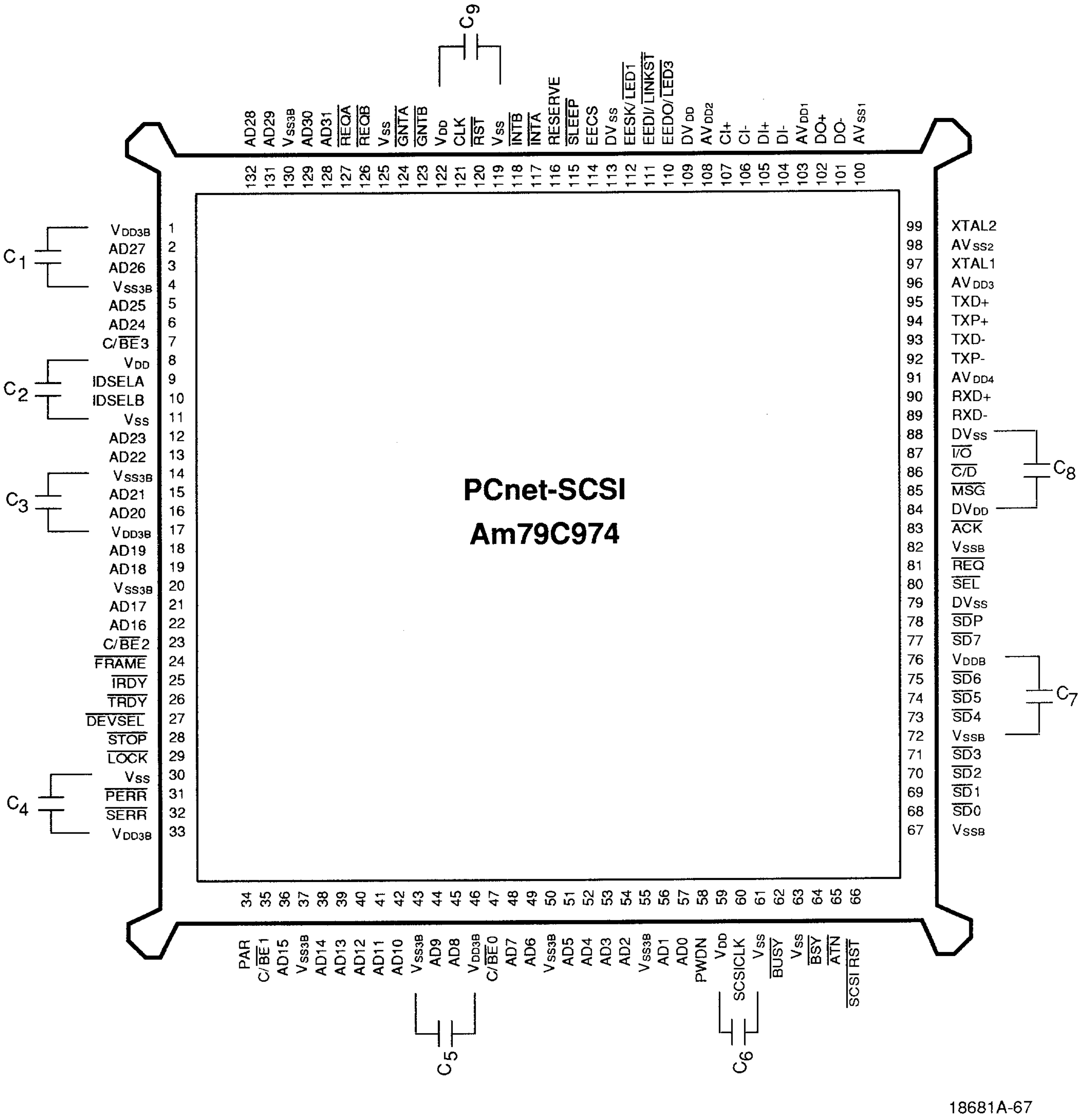
- A 40 MHz crystal oscillator must be used in order to have a 10 MB/s SCSI data rate. Use of this 40 MHz crystal oscillator, which drives the SCSI CLK pin on the Am79C974, introduces the possibility of unwanted high frequency components, including harmonics, coupling with Am79C974 signals. To help prevent this, a resistor should be placed in series with the oscillator to form a low pass filter with the input capacitance (10 pF) of the SCSI CLK pin. This filter reduces the edge rate of the clock waveform, increasing both rise and fall times by 2 ns. This removes higher frequencies, specifically harmonics from the crystal oscillator waveform and thus reduces the amount of noise introduced into the Am79C974.
- A ferrite bead, which is essentially an inductor, may be used with the oscillator to prevent coupling of higher frequencies with DC power supply signals. The ferrite bead blocks high frequency noise while acting like a short to the DC components.

- From an EMI standpoint, SCSI-2 high-density cables should be used instead of a SCSI-1 cables. Unlike the SCSI-1 flat ribbon cable, the SCSI-2 cable is electrically more substantial. It is shielded and signal wires are strategically placed for better signal travel.

## Decoupling Methods

As stated in Appendix C, decoupling capacitors should be used across the $V_{DD}$ and $V_{SS}$ pins on the motherboard There are pairs of $V_{DD}$ and $V_{SS}$ pins on the Am79C974 that should each have their own decoupling capacitor. The following decoupling method should be used for the $V_{DD}/V_{SS}$ pairs:

- Connect the capacitor directly between a $V_{DD}/V_{SS}$ pair of the Am79C974 so that it sits on the component side of the board. This configuration will allow the capacitor to filter undesired high frequency components directly at the Am79C974 and not only at the power planes. This not only minimizes the noise on the power planes that the chip sees, but it also filters any noise generated by the chip before it reaches the power planes. The leads to each end of the capacitor should be wide and may contain several feed-throughs to the $V_{DD}$ and $V_{SS}$ planes to reduce the inductance present. Refer to Figure E-5.



*where C1 – C9 are decoupling capacitors.*

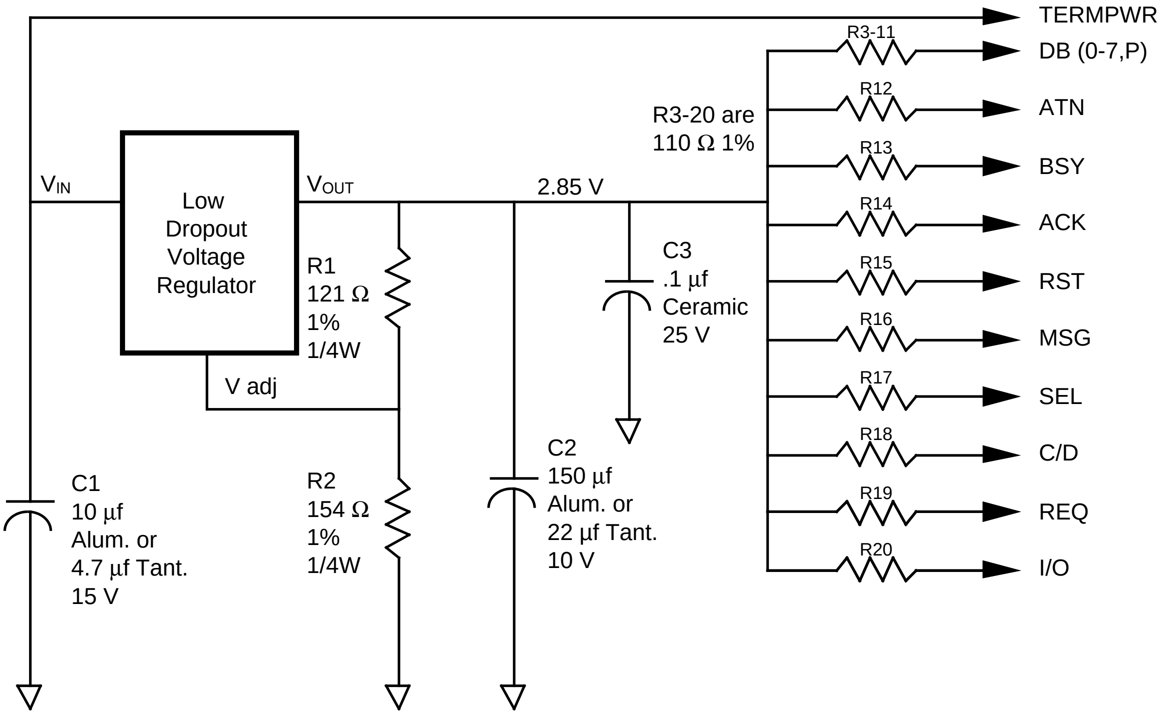
**Figure E-5. Decoupling Capacitor Placement**

Decoupling methods which are NOT recommended include:

- connecting a capacitor only between the $V_{DD}$ and $V_{SS}$ planes so that it sits on the component side of the board. This doesn't allow the capacitor to reduce noise directly at the chip, but it does allow for a reduction of power plane noise and of board components. (capacitors).
- connecting a capacitor only between the $V_{DD}$ and $V_{SS}$ planes so that it sits on the solder side of the board. This configuration also doesn't allow direct decoupling at the chip. It does save board space, but it requires an extra manufacturing step.

## TERMINATION CONSIDERATIONS

The use of active or regulated termination for terminators on the motherboard is recommended (see Figure E-6), while external SCSI terminators can be used to terminate other parts of the SCSI bus. Terminators must match the impedance seen by a signal at the end of the SCSI bus to the characteristic impedance of the SCSI bus. This impedance is typically 84 +/– 12 Ω, but can vary greatly with PC board characteristics and cabling.



18681A-68

**Figure E-6. Regulated Termination**

## Termination

There are three general termination schemes that apply to motherboard or host adapter setups when using regulated terminators. Each scheme recognizes that the SCSI bus must be terminated on both ends. Therefore, as more components and peripherals are added to the bus, terminators must be relocated accordingly. Each scheme is also based on an ideal routing situation, that is one where the internal peripherals, external peripherals and the Am79C974 chip are connected by a linear SCSI bus. See Figure E-1.

### Scheme #1

In this case, the system uses only SCSI internal peripherals. The regulated terminators on board should be activated. Peripherals can be added to the bus by connecting them to a 50-pin ribbon cable. A SCSI terminator should be attached to the last peripheral to terminate the other end of the bus.

### Scheme #2

In this case, the system uses only external SCSI peripherals. As in Scheme #1, the on-board regulated terminators should be activated. In this scheme, a 50-pin high density SCSI-2 cable should connect the external port on the motherboard or host adapter to the first external peripheral. Peripherals may be added with more cables and the last peripheral should be terminated.

### Scheme #3

In this case, both internal and external SCSI peripherals are used. The regulated terminators should be deactivated (since the Am79C974 will sit in the middle of the SCSI bus). This may be accomplished through hardware or software. The hardware approach involves developing a mechanism to detect peripherals connected to the SCSI bus. This mechanism must then activate a signal to turn the regulated terminators on or off accordingly. The software approach involves having the user tell the system interactively that the regulated terminators should be turned on or off. Care should be taken to ensure that each end of the Bus is terminated.

## OTHER CONSIDERATIONS

The following are motherboard considerations for routing and layout. They should be taken into account along with SCSI considerations.

## Motherboards

1. High speed signals should be referenced only to the ground plane or exclusively to one of the power planes. If not, the power planes should be decoupled.
2. For a PCI CLK of 33 MHz, the maximum round trip time of any shared mother board signal should be 10 ns.

# Designing a Single Motherboard for AMD PCI Family

Three devices in the AMD PCI family, the Am53C974, the Am79C970, and the Am79C794, were designed with very similar pin assignments so that a single motherboard design  could be used for three different products:  one with a built-in SCSI controller, one with a built-in Ethernet controller, and one with both a SCSI controller and an Ethernet controller. This discussion describes some implementation details.

## Pin Out Differences

The table below shows the pins that are not the same on the three devices. All pins that are shown as NC are not bonded to the die inside the device package. These pins can be driven by any signal without affecting the operation of the device. Those pins that are shown in the PCI family data sheets as RESERVED are connected to test logic on the die. These pins are active only during stand-alone chip testing, and therefore they may also be connected to signals on the board without affecting the operation of the device. On the other hand, pin 116 which is shown as RESERVED-DNC (for Do Not Connect) must not be connected to anything on the board.

| Pin Numbers | Am79C970 (Ethernet) Function | Am53C974 (SCSI) Function | Am79C974 (Combination) Function |
|---|---|---|---|
| 9 | RESERVED | IDSEL | IDSELA |
| 10 | IDSEL | NC | IDSELB |
| 58 | RESERVED | PWDN | PWDN |
| 60 | NC | SCSICLK1 | SCSICLK |
| 68, 69, 70, 71, 73, 74, 75, 77, 78 | NC | SCSI data & parity bus | SCSI data & parity bus |
| 64, 65, 66, 80, 81, 83, 85, 86, 87 | NC | SCSI control lines | SCSI control lines |
| 89, 90, 92, 93, 94, 95 | 10Base-T lines | NC | 10Base-T lines |
| 97 | XTAL1 | SCSICLK2 | SCSICLK |
| 99 | XTAL2 | NC | XTAL2 |
| 101, 102, 104, 105, 106, 107 | AUI lines | NC | AUI lines |
| 110, 111, 112, 114 | EEPROM interface & LED pins | NC | EEPROM interface & LED pins |
| 115 | $\overline{\text{SLEEP}}$ | NC | $\overline{\text{SLEEP}}$ |
| 116 | RESERVED_DNC | RESERVED_DNC | RESERVED_DNC |
| 117 | $\overline{\text{INTA}}$ | $\overline{\text{INTA}}$ | $\overline{\text{INTA}}$ |
| 118 | NC | NC | $\overline{\text{INTB}}$ |
| 126 | $\overline{\text{REQ}}$ | NC | $\overline{\text{REQB}}$ |
| 127 | RESERVED | $\overline{\text{REQ}}$ | $\overline{\text{REQA}}$ |
| 123 | $\overline{\text{GNT}}$ | NC | $\overline{\text{GNTB}}$ |
| 124 | NC | $\overline{\text{GNT}}$ | $\overline{\text{GNTA}}$ |

## INTx (Pins 117 and 118)

The Am79C970 and Am53C974 devices have only one interrupt output, which is connected to pin 117. The Am79C974, on the other hand, has two interrupt outputs:  $\overline{\text{INTA}}$, which is used for SCSI interrupts, is connected to pin 117; while $\overline{\text{INTB}}$, which is used for Ethernet interrupts, is connected to pin 118. The motherboard can connect these pins directly to dedicated inputs to its interrupt controller, or it can connect them to multiplex logic that can map these outputs to interrupt controller inputs by means of software.

## $\overline{\text{REQ}}$ and $\overline{\text{GNT}}$ (pins 123, 124, 126, and 127)

The Am53C974 uses pins 127 and 124 ($\overline{\text{REQ}}$ and $\overline{\text{GNT}}$) for bus master arbitration, while the Am79C790 uses pins 126 and 123. The Am79C974 uses pins 127 and 124 ($\overline{\text{REQA}}$ and $\overline{\text{GNTA}}$) for bus arbitration from the SCSI controller, and it uses pins 126 and 123 ($\overline{\text{REQB}}$ and $\overline{\text{GNTB}}$) for arbitration for the Ethernet controller. The motherboard can connect both pairs of signals to its bus arbitration logic through jumpers or zero Ohm series resistors. One pair of resistors or jumpers would be omitted for a SCSI only product, and the other pair would be omitted for an Ethernet only product. Both pairs would be inserted for a product with both SCSI and Ethernet.

If the motherboard uses a PCI controller that allows a limited number of bus masters, you may have to omit one PCI slot or define one slot to be slave only for a product with both SCSI and Ethernet on the motherboard.

## IDSEL (Pins 9 and 10)

The Am53C974 uses pin 9 for IDSEL, which is a chip select for the PCI configuration space, while the Am79C790 uses pin 10 for IDSEL. The Am79C974, which has two separate configuration spaces, uses pin 9 (IDSELA) to access the configuration space for the SCSI controller and pin 10 (IDSELB) for the Ethernet controller. IDSEL is used only during configuration cycles, which are defined by the command on the C/$\overline{\text{BE}}$ lines during the address phase.

A typical way to generate the IDSEL signals on a motherboard is to connect a separate address line to each of the IDSEL pins of the various PCI devices and expansion slots. Since a typical PCI motherboard will not have more than 10 PCI devices and slots, two of the high order address lines can be connected directly to pins 9 and 10 of the AMD device. The Am79C970 will respond only to configuration cycles when pin 10 is active and will ignore configuration cycles when pin 9 is active. The Am53C794 will respond only to configuration cycles when pin 9 is active.

## Clocks (Pins 60, 97, and 99)

The Am53C974 requires two clock inputs: SCSICLK1 (pin 60) and SCSICLK2 (pin 97). These clock inputs can be driven by the same clock source. The Am79C790 requires either a crystal connected to the XTAL1 and XTAL2 pins (97 and 99) or an external oscillator connected to XTAL1. The Am79C974 requires both a clock input for SCSICLK (pin 60) and either a crystal connected to the XTAL1 and XTAL2 pins (97 and 99) or an external oscillator connected to XTAL1.

To satisfy all of these requirements, the motherboard can connect pin 60 to pin 97 through a jumper or zero Ohm resistor. For the SCSI only product, the oscillator or crystal should be omitted and the jumper inserted. For the Ethernet-only product the oscillator or crystal should be inserted and the jumper omitted. It does not matter whether or not the SCSI oscillator is included. For the combination SCSI plus Ethernet product, the crystal and the SCSI oscillator should be included, and the jumper must be omitted.

Figure F-1 illustrates these connections.

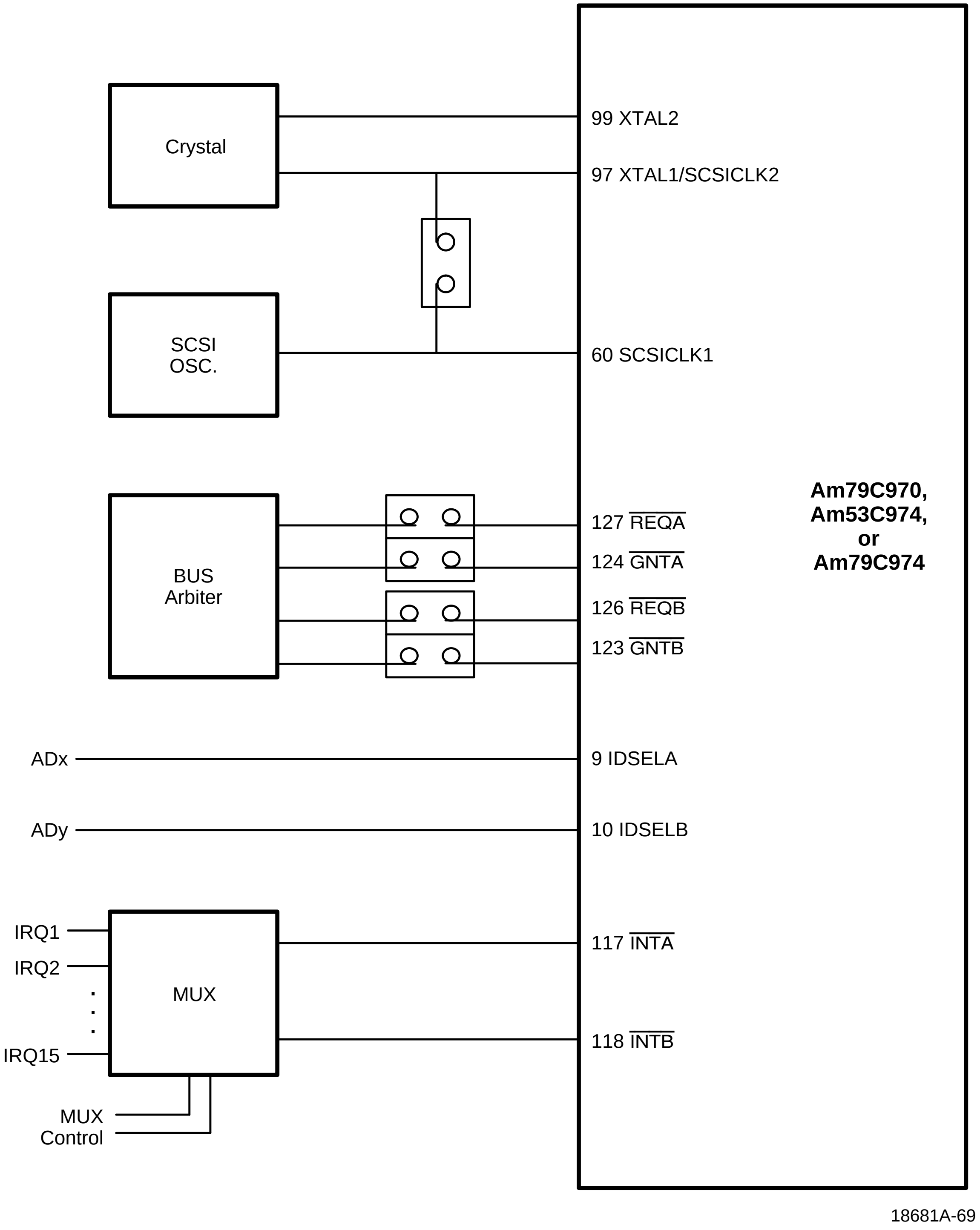

Crystal
SCSI OSC.
BUS Arbiter
MUX
99 XTAL2
97 XTAL1/SCSICLK2
60 SCSICLK1
127 $\overline{\text{REQA}}$
124 $\overline{\text{GNTA}}$
126 $\overline{\text{REQB}}$
123 $\overline{\text{GNTB}}$
9 IDSELA
10 IDSELB
117 $\overline{\text{INTA}}$
118 $\overline{\text{INTB}}$
Am79C970, Am53C974, or Am79C974
ADx
ADy
IRQ1
IRQ2
IRQ15
MUX Control
18681A-69

**Figure F-1. PCI Family Connections**

For information on additional SCSI software products contact:

**Sequoia Advanced Technologies, Inc.**
(415) 459-7978
(415) 459-7988 FAX
71322, 1020 CompuServe ID

**Trademarks**

AMENDMENT

# Am79C974

## PCnet$^{TM}$-SCSI Combination Ethernet and SCSI Controller for PCI Systems

Advanced
Micro
Devices

This amendment corrects a few minor inaccuracies and adds or clarifies a few sections. Minor corrections should be made on the existing data sheets. However, for ease of use, pages with more than a word or two of corrections are printed with this amendment.

### Details:

**Page 12**
**(Reprinted as page 5 of this amendment)**

CONNECTION DIAGRAM

- Change the names of the following power pins:
  - — Change from $V_{DD3B}$ to **$V_{DDB}$.**
  - — Change from $V_{SS3B}$ to **$V_{SSB}$.**
  - — Change from $V_{DDB}$ to **$V_{DDBS}$.**
  - — Change from $V_{SSB}$ to **$V_{SSBS}$.**

**Pages 14–15**
**(Reprinted as pages 6, 7 of this amendment)**

PIN DESIGNATIONS, Listed by Pin Number and Pin Name

- Change the names of the following power pins:
  - — Change from $V_{DD3B}$ to **$V_{DDB}$.**
  - — Change from $V_{SS3B}$ to **$V_{SSB}$.**
  - — Change from $V_{DDB}$ to **$V_{DDBS}$.**
  - — Change from $V_{SSB}$ to **$V_{SSBS}$.**

**Page 17**
**(Reprinted as page 8 of this amendment)**

PIN DESIGNATIONS, Quick Reference Pin Description

- The number of each type of power pin is incorrect. The numbers should be:
  - — Change from $V_{DD}$ to **$V_{DD}$/$DV_{DD}$, Digital Power, 5 pins**
  - — Change from $V_{DDB}$/$V_{DD3B}$ to **$V_{DDB}$/$V_{DDBS}$, I/O Buffer Power, 5 pins**
  - — Change from $V_{SS}$ to **$V_{SS}$/$DV_{SS}$, Digital Ground, 9 pins**
  - — Change from $V_{SSB}$/$V_{SS3B}$ to **$V_{SSB}$/$V_{SSBS}$, I/O Buffer Ground, 11 pins**

PIN DESIGNATIONS, Listed By Driver Type

Change the $I_{OH}$ value for both TS3 and TS6 from –0.4 mA to **–2.0 mA**.

Publication# **18681** Rev. **A** Amendment **/1**
Issue Date: **April 1994**

**Page 19**
**(Reprinted as page 9 of this amendment)**

PIN DESCRIPTION, $\overline{\text{GNTA}}$

Add after the second paragraph:

The Am79C974 supports bus parking. When the PCI bus is idle and the system arbiter asserts $\overline{\text{GNTA}}$ without an active $\overline{\text{REQA}}$ from the Am79C974 controller, the Am79C794 will actively drive the AD[31:00], C/$\overline{\text{BE}}$[3:0], and PAR lines.

**Page 20**

Note that $\overline{\text{INTA}}$ and $\overline{\text{INTB}}$ are both open drain pins.

**Page 24**

Note that $\overline{\text{ATN}}$ is open drain.

**Page 25**

- The number of each type of power pin is incorrect. The numbers should be:
  — $V_{DD}$/$DV_{DD}$, Digital Power, 5 pins
  — $V_{DDB}$/$V_{DDBS}$, I/O Buffer Power, 5 pins
  — $V_{SS}$/$DV_{SS}$ Digital Ground, 9 pins
  — $V_{SSB}$/$V_{SSBS}$, I/O Buffer Ground, 11 pins

**Page 30**

The second sentence should read, “It is a single cycle, non-burst 8-bit, 16-bit, or 32-bit transfer which is initiated by the host CPU.” (The original omitted “8-bit”).

**Page 32**
**(Reprinted as page 10 of this amendment)**

In Figure 6, data on the AD lines should be driven during clock 6 and the second instance of PAR should be driven during clock 7. In each case this is one cycle earlier than what is shown in the figure.

**Page 35**

Line 2, change “type 15” to **“type 14”**.

**Page 39**
**(Reprinted as page 11 of this amendment)**

Replace the last two sentences with the following:

“For SCSI, when target aborts, $\overline{\text{INTA}}$ will not be asserted, but the ABORT bit (bit 2 of the DMA status register at offset 54h) is set. For either Ethernet or SCSI a target abort causes RTABORT (bit 12) of the status register in the appropriate PCI configuration space to be set.” (The original stated that $\overline{\text{INTA}}$ will be asserted.)

**Page 42**
**(Reprinted as page 12 of this amendment)**

Replace the last sentence with the following:

“For SCSI, when the master aborts, $\overline{\text{INTA}}$ will not be asserted, but the ABORT bit (bit 2 of the DMA status register at offset 54h) is set. For either Ethernet or SCSI a master abort causes RMABORT (bit 13) of the status register in the appropriate PCI configuration space to be set.” (The original stated that $\overline{\text{INTA}}$ will be asserted.)

**Page 44**

Change SPRINTEN to **LAPPEN** in lines 12 and 13.

**Page 54**

Line 17, change 100% to **10%**.

**Page 59**

Line 12 should read, “When the Am79C974 controller samples its IDSELA or IDSELB input ...” (The original ommitted “IDSELA”).

**Page 62**

Column 2, line 30, change “lower two bytes” to “**upper two bytes**”.

**Page 71**

Line 15, change “DRCBC” to “**DRCVPA**”.

**Page 76**
**(Reprinted as pages 13 and 14 of the amendment)**

“DMA BLAST command” section has been extensively revised.

**Page 77**
**(Reprinted as page 15 of this amendment)**

The description of the CMD1–0 bits at the end of column 2 should read, “These two bits are encoded to represent four commands: IDLE, BLAST, START, and ABORT.” (The BLAST command was omitted in the original.)

In the table at the bottom, the description of the IDLE command should read, “Resets the DMA block to the IDLE state. Stops any current transfer. Does not affect status bits or cause an interrupt.”

**Pages 78–80**
**(Reprinted as pages 16–18 of this amendment)**

“DMA Scatter-Gather Operation (4K aligned elements)” section has been extensively revised.

**Page 81**

Delete lines 19 and 20, which read, “The DMA transfer request was aborted (ABORT command, or the Master or Target Abort).”

**Page 92**
ABSOLUTE MAXIMUM RATINGS

Line 4: Add “or $V_{SSBS}$ or $DV_{SS}$” to read “Supply Voltage to $AV_{SS}$ or $V_{SSB}$ or $V_{SSBS}$ or $DV_{SS}$”

Line 5: Add “$V_{DDBS}$, $DV_{DD}$” to read “($AV_{DD}$, $V_{DD}$, $V_{DDB}$, $V_{DDBS}$, $DV_{DD}$)”

**Page 92**
OPERATING RANGES

Line 3: Add “$V_{DDBS}$, $DV_{DD}$” to read “($AV_{DD}$, $V_{DD}$, $V_{DDB}$, $V_{DDBS}$, $DV_{DD}$)”

After Line 7: Add “or $V_{SSBS}$ –0.5 V $\leq V_{IN} \leq V_{DDBS}$ +0.5 V” and “or $DV_{SS}$ –0.5 V $\leq V_{IN} \leq DV_{DD}$ +0.5 V”

In the DC Characteristics table, $V_{OL}$ (max) for those pins to which $I_{OL1}$ or $I_{OL2}$ applies is 0.55 V (to comply with the PCI specification) rather than 0.45 V. Also, $V_{OL}$ (max) for those pins to which $I_{OL3}$ applies is 0.4 V rather than 0.45 V.

**Page 94**

In the DC Characteristics table, delete row 5, $V_{SOL1}$. Change the parameter symbol for row 6 from $V_{SOL2}$ to $V_{OL}$. Add $\overline{SD}$[7:0] and $\overline{SD}$P to the parameter description for row 6.

**Page 95**

In the DC Characteristics table, move “PWDN” from the list of pin names in row 1 as a separate item from PCI input exceptions. The entry in row 1, column 3 should read, “All SCSI inputs. All Ethernet Inputs. PWDN. All PCI inputs except IDSEL.”

**Page 113**

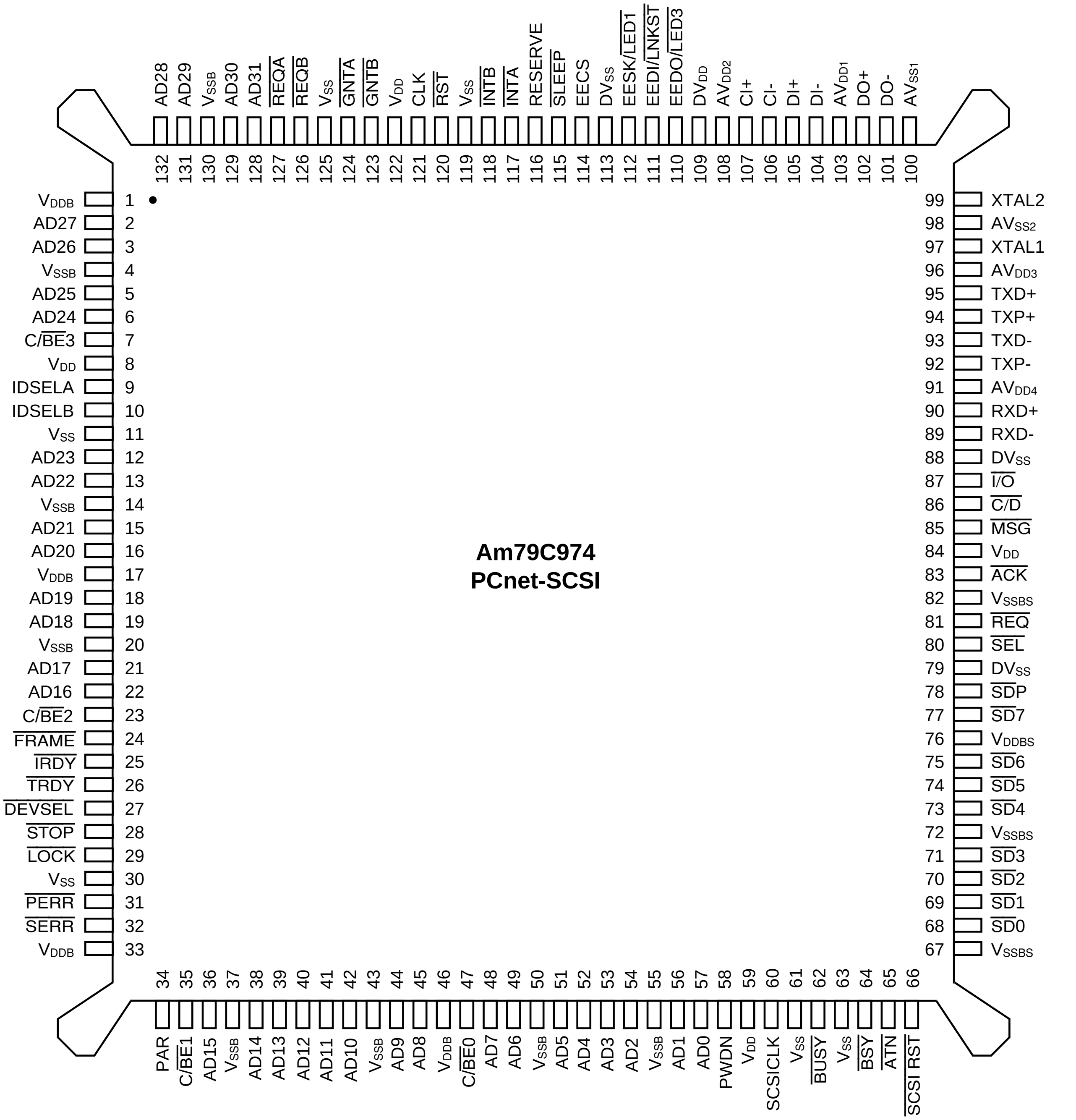
In the Ethernet PCI Configuration Registers table, the default value for the Interrupt pin (row 13, column 5) should be 02h, not 01h.

**Page 129**
**(Reprinted as page 19 of this amendment)**

Figure E-5 Decoupling Capacitor Placement

- Change the names of the following power pins:
  - — Change from $V_{DD3B}$ to $\mathbf{V_{DDB}}$**.**
  - — Change from $V_{SS3B}$ to $\mathbf{V_{SSB}}$**.**
  - — Change from $V_{DDB}$ to $\mathbf{V_{DDBS}}$**.**
  - — Change from $V_{SSB}$ to $\mathbf{V_{SSBS}}$**.**

## CONNECTION DIAGRAM



18681A/1-3

*Pin 1 is marked for orientation.*

*RESERVE = Don’t Connect.*

## PIN DESIGNATIONS

### Listed by Pin Number

| Pin No. | Pin Name | Pin No. | Pin Name | Pin No. | Pin Name | Pin No. | Pin Name |
|---|---|---|---|---|---|---|---|
| 1 | $V_{DDB}$ | 34 | PAR | 67 | $V_{SSBS}$ | 100 | $AV_{SS1}$ |
| 2 | AD27 | 35 | C/$\overline{\text{BE}}$1 | 68 | $\overline{\text{SD}}$0 | 101 | DO– |
| 3 | AD26 | 36 | AD15 | 69 | $\overline{\text{SD}}$1 | 102 | DO+ |
| 4 | $V_{SSB}$ | 37 | $V_{SSB}$ | 70 | $\overline{\text{SD}}$2 | 103 | $AV_{DD1}$ |
| 5 | AD25 | 38 | AD14 | 71 | $\overline{\text{SD}}$3 | 104 | DI– |
| 6 | AD24 | 39 | AD13 | 72 | $V_{SSBS}$ | 105 | DI+ |
| 7 | C/$\overline{\text{BE}}$3 | 40 | AD12 | 73 | $\overline{\text{SD}}$4 | 106 | CI– |
| 8 | $V_{DD}$ | 41 | AD11 | 74 | $\overline{\text{SD}}$5 | 107 | CI+ |
| 9 | IDSELA | 42 | AD10 | 75 | $\overline{\text{SD}}$6 | 108 | $AV_{DD2}$ |
| 10 | IDSELB | 43 | $V_{SSB}$ | 76 | $V_{DDBS}$ | 109 | $DV_{DD}$ |
| 11 | $V_{SS}$ | 44 | AD9 | 77 | $\overline{\text{SD}}$7 | 110 | EEDO/$\overline{\text{LED3}}$ |
| 12 | AD23 | 45 | AD8 | 78 | $\overline{\text{SD}}$P | 111 | EEDI/$\overline{\text{LNKST}}$ |
| 13 | AD22 | 46 | $V_{DDB}$ | 79 | $DV_{SS}$ | 112 | EESK/$\overline{\text{LED1}}$ |
| 14 | $V_{SSB}$ | 47 | C/$\overline{\text{BE}}$0 | 80 | $\overline{\text{SEL}}$ | 113 | $DV_{SS}$ |
| 15 | AD21 | 48 | AD7 | 81 | $\overline{\text{REQ}}$ | 114 | EECS |
| 16 | AD20 | 49 | AD6 | 82 | $V_{SSBS}$ | 115 | $\overline{\text{SLEEP}}$ |
| 17 | $V_{DDB}$ | 50 | $V_{SSB}$ | 83 | $\overline{\text{ACK}}$ | 116 | RESERVE |
| 18 | AD19 | 51 | AD5 | 84 | $DV_{DD}$ | 117 | $\overline{\text{INTA}}$ |
| 19 | AD18 | 52 | AD4 | 85 | $\overline{\text{MSG}}$ | 118 | $\overline{\text{INTB}}$ |
| 20 | $V_{SSB}$ | 53 | AD3 | 86 | $\overline{\text{C/D}}$ | 119 | $V_{SS}$ |
| 21 | AD17 | 54 | AD2 | 87 | $\overline{\text{I/O}}$ | 120 | $\overline{\text{RST}}$ |
| 22 | AD16 | 55 | $V_{SSB}$ | 88 | $DV_{SS}$ | 121 | CLK |
| 23 | C/$\overline{\text{BE}}$2 | 56 | AD1 | 89 | RXD– | 122 | $V_{DD}$ |
| 24 | $\overline{\text{FRAME}}$ | 57 | AD0 | 90 | RXD+ | 123 | $\overline{\text{GNTB}}$ |
| 25 | $\overline{\text{IRDY}}$ | 58 | PWDN | 91 | $AV_{DD4}$ | 124 | $\overline{\text{GNTA}}$ |
| 26 | $\overline{\text{TRDY}}$ | 59 | $V_{DD}$ | 92 | TXP– | 125 | $V_{SS}$ |
| 27 | $\overline{\text{DEVSEL}}$ | 60 | SCSICLK | 93 | TXD– | 126 | $\overline{\text{REQB}}$ |
| 28 | $\overline{\text{STOP}}$ | 61 | $V_{SS}$ | 94 | TXP+ | 127 | $\overline{\text{REQA}}$ |
| 29 | $\overline{\text{LOCK}}$ | 62 | $\overline{\text{BUSY}}$ | 95 | TXD+ | 128 | AD31 |
| 30 | $V_{SS}$ | 63 | $V_{SS}$ | 96 | $AV_{DD3}$ | 129 | AD30 |
| 31 | $\overline{\text{PERR}}$ | 64 | $\overline{\text{BSY}}$ | 97 | XTAL1 | 130 | $V_{SSB}$ |
| 32 | $\overline{\text{SERR}}$ | 65 | $\overline{\text{ATN}}$ | 98 | $AV_{SS2}$ | 131 | AD29 |
| 33 | $V_{DDB}$ | 66 | $\overline{\text{SCSI RST}}$ | 99 | XTAL2 | 132 | AD28 |

## PIN DESIGNATIONS

### Listed by Pin Name

| Pin Name | Pin No. | Pin Name | Pin No. | Pin Name | Pin No. | Pin Name | Pin No. |
|---|---|---|---|---|---|---|---|
| $\overline{\text{ACK}}$ | 83 | $\overline{\text{ATN}}$ | 65 | $\overline{\text{GNTB}}$ | 123 | $\overline{\text{STOP}}$ | 28 |
| AD0 | 57 | $AV_{DD1}$ | 103 | IDSELA | 9 | $\overline{\text{TRDY}}$ | 26 |
| AD1 | 56 | $AV_{DD2}$ | 108 | IDSEL | 10 | XTAL1 | 97 |
| AD2 | 54 | $AV_{DD3}$ | 96 | $\overline{\text{INTA}}$ | 117 | XTAL2 | 99 |
| AD3 | 53 | $AV_{DD4}$ | 91 | $\overline{\text{INTB}}$ | 118 | TXD– | 93 |
| AD4 | 52 | $AV_{SS1}$ | 100 | $\overline{\text{I/O}}$ | 87 | TXD+ | 95 |
| AD5 | 51 | $AV_{SS2}$ | 98 | $\overline{\text{IRDY}}$ | 25 | TXP– | 92 |
| AD6 | 49 | $\overline{\text{BSY}}$ | 64 | $\overline{\text{LOCK}}$ | 29 | TXP+ | 94 |
| AD7 | 48 | $\overline{\text{BUSY}}$ | 62 | $\overline{\text{MSG}}$ | 85 | $V_{DD}$ | 8 |
| AD8 | 45 | C/$\overline{\text{BE}}$0 | 47 | PAR | 34 | $V_{DD}$ | 59 |
| AD9 | 44 | C/$\overline{\text{BE}}$1 | 35 | $\overline{\text{PERR}}$ | 31 | $V_{DD}$ | 122 |
| AD10 | 42 | C/$\overline{\text{BE}}$2 | 23 | PWDN | 58 | $V_{DDB}$ | 1 |
| AD11 | 41 | C/$\overline{\text{BE}}$3 | 7 | $\overline{\text{REQ}}$ | 81 | $V_{DDB}$ | 17 |
| AD12 | 40 | $\overline{\text{C/D}}$ | 86 | $\overline{\text{REQA}}$ | 127 | $V_{DDB}$ | 33 |
| AD13 | 39 | CLK | 121 | $\overline{\text{REQB}}$ | 126 | $V_{DDB}$ | 46 |
| AD14 | 38 | CI– | 106 | RESERVE | 116 | $V_{DDBS}$ | 76 |
| AD15 | 36 | CI+ | 107 | $\overline{\text{RST}}$ | 120 | $V_{SS}$ | 11 |
| AD16 | 22 | $\overline{\text{DEVSEL}}$ | 27 | RXD– | 89 | $V_{SS}$ | 30 |
| AD17 | 21 | DI– | 104 | RXD+ | 90 | $V_{SS}$ | 61 |
| AD18 | 19 | DI+ | 105 | SCSICLK | 60 | $V_{SS}$ | 63 |
| AD19 | 18 | DO– | 101 | $\overline{\text{SCSI RST}}$ | 66 | $V_{SS}$ | 119 |
| AD20 | 16 | DO+ | 102 | $\overline{\text{SD0}}$ | 68 | $V_{SS}$ | 125 |
| AD21 | 15 | $DV_{DD}$ | 84 | $\overline{\text{SD1}}$ | 69 | $V_{SSB}$ | 4 |
| AD22 | 13 | $DV_{DD}$ | 109 | $\overline{\text{SD2}}$ | 70 | $V_{SSB}$ | 14 |
| AD23 | 12 | $DV_{SS}$ | 79 | $\overline{\text{SD3}}$ | 71 | $V_{SSB}$ | 20 |
| AD24 | 6 | $DV_{SS}$ | 88 | $\overline{\text{SD4}}$ | 73 | $V_{SSB}$ | 37 |
| AD25 | 5 | $DV_{SS}$ | 113 | $\overline{\text{SD5}}$ | 74 | $V_{SSB}$ | 43 |
| AD26 | 3 | EECS | 114 | $\overline{\text{SD6}}$ | 75 | $V_{SSB}$ | 50 |
| AD27 | 2 | EEDI/$\overline{\text{LNKST}}$ | 111 | $\overline{\text{SD7}}$ | 77 | $V_{SSB}$ | 55 |
| AD28 | 132 | EEDO/$\overline{\text{LED3}}$ | 110 | $\overline{\text{SDP}}$ | 78 | $V_{SSB}$ | 130 |
| AD29 | 131 | EESK/$\overline{\text{LED1}}$ | 112 | $\overline{\text{SEL}}$ | 80 | $V_{SSBS}$ | 67 |
| AD30 | 129 | $\overline{\text{FRAME}}$ | 24 | $\overline{\text{SERR}}$ | 32 | $V_{SSBS}$ | 72 |
| AD31 | 128 | $\overline{\text{GNTA}}$ | 124 | $\overline{\text{SLEEP}}$ | 115 | $V_{SSBS}$ | 82 |

## PIN DESIGNATIONS (continued)

### Quick Reference Pin Description

| Pin Name | Description | Type | Driver | # Pins |
|---|---|---|---|---|
| **SCSI SPECIFIC**<br>**SCSI Interface** | | | | |
| $\overline{\text{SD}}$ [7:0] | SCSI Data | IO | OD48 | 8 |
| $\overline{\text{SDP}}$ | SCSI Data Parity | IO | OD48 | 1 |
| $\overline{\text{MSG}}$ | Message | I | | 1 |
| $\overline{\text{C/D}}$ | Command/Data | I | | 1 |
| $\overline{\text{I/O}}$ | Input/Output | I | | 1 |
| $\overline{\text{ATN}}$ | Attention | O | OD48 | 1 |
| $\overline{\text{BSY}}$ | Busy | IO | OD48 | 1 |
| $\overline{\text{SEL}}$ | Select | IO | OD48 | 1 |
| $\overline{\text{SCSI RST}}$ | SCSI Bus Reset | IO | OD48 | 1 |
| $\overline{\text{REQ}}$ | Request | I | | 1 |
| $\overline{\text{ACK}}$ | Acknowledge | O | OD48 | 1 |
| **Miscellaneous** | | | | |
| SCSI CLK | SCSI Core Clock | I | | 1 |
| RESERVE | Reserved, DO NOT CONNECT | I | | 1 |
| **Power Management** | | | | |
| PWDN | Power Down Indicator | I | | 1 |
| **Test Interface** | | | | |
| $\overline{\text{BUSY}}$ | NAND Tree Test Output | O | O3 | 1 |
| **Power Supplies** | | | | |
| $AV_{DD}$ | Analog Power | P | NA | 4 |
| $AV_{SS}$ | Analog Ground | P | NA | 2 |
| $V_{DD}$, $DV_{DD}$ | Digital Power | P | NA | 5 |
| $V_{SS}$, $DV_{SS}$ | Digital Ground | P | NA | 9 |
| $V_{DDB}$, $V_{DDBS}$ | I/O Buffer Power | P | NA | 5 |
| $V_{SSB}$, $V_{SSBS}$ | I/O Buffer Ground | P | NA | 11 |

### Listed by Driver Type

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

| Name | Type | $I_{OL}$ (mA) | $I_{OH}$ (mA) | pF |
|---|---|---|---|---|
| TS3 | Tri-State™ | 3 | –2.0 | 50 |
| TS6 | Tri-State | 6 | –2.0 | 50 |
| O3 | Totem Pole | 3 | –0.4 | 50 |
| O6 | Totem Pole | 6 | –0.4 | 50 |
| O8 | Totem Pole | 8 | –0.4 | 50 |
| OD6 | Open Drain | 6 | NA | 50 |
| OD48 | Open Drain | 48 | NA | — |
| LED | LED | 12 | –0.4 | 50 |

# PIN DESCRIPTION

## PCI Bus Interface

### AD[31:00]

**Address and Data**
**Input/Output, Active High**

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

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

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

When RST is active, AD[31:0] are inputs for NAND tree testing.

### C/BE [3:0]

**Bus Command and Byte Enables**
**Input/Output, Active Low**

These signals are multiplexed on the same PCI pins. During the address phase of the transaction, C/BE[3:0] define the bus command. During the data phase C/BE[3:0] are used as Byte Enables. The Byte Enables define which physical byte lanes carry meaningful data. C/BE0 applies to byte 0 (AD[07:00]) and C/BE3 applies to byte 3 (AD[31:24]). The function of the Byte Enables is independent of the byte ordering mode (CSR3, bit 2).

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

### CLK

**Clock**
**Input**

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

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

### DEVSEL

**Device Select**
**Input/Output, Active Low**

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

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

### FRAME

**Cycle Frame**
**Input/Output, Active Low**

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

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

### GNTA

**Bus Grant**
**Input, Active Low**

This signal indicates that the access to the bus has been granted to the Am79C974’s SCSI controller.

The Am79C974 supports bus parking. When the PCI bus is idle and the system arbiter asserts GNTA without an active REQA from the Am79C974 controller, the Am79C974 will actively drive the AD[31:00], C/BE[3:0], and PAR lines.

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

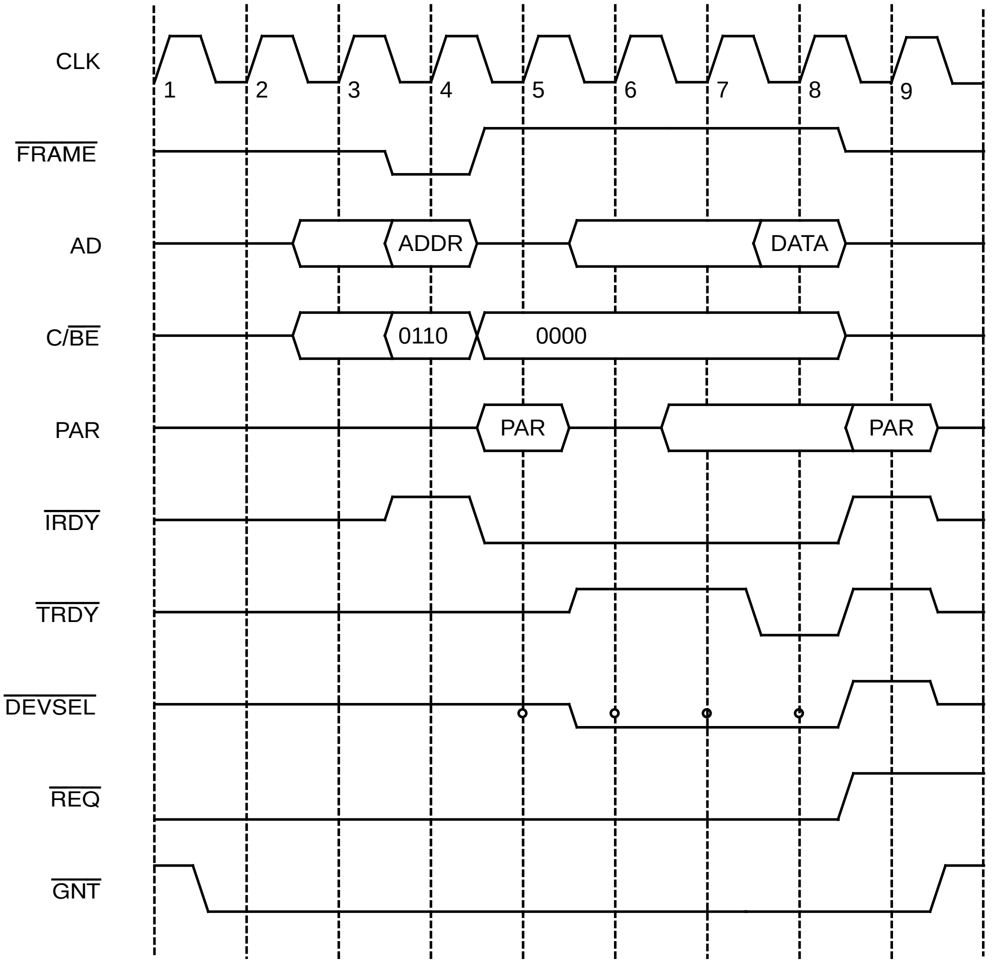
### Bus Master DMA Transfers

There are four primary types of DMA transfers. The Am79C974 controller uses non-burst as well as burst cycles for read and write access to the main memory.

### Basic Non-Burst Read Cycles

All Am79C974 controller non-burst read accesses are of the PCI command type Memory Read (type 6). Note that during all non-burst read operations, the Am79C974 controller will always activate all byte enables, even though some byte lanes may not contain valid data as indicated by a buffer pointer value. In such instances, the Am79C974 controller will internally discard unneeded bytes.

Figure 6 shows a typical non-burst read access. The Am79C974 controller asserts $\overline{\text{IRDY}}$ at clock 5 immediately after the address phase and starts sampling $\overline{\text{DEVSEL}}$. The target extends the cycle by asserting $\overline{\text{DEVSEL}}$ not until clock 6. Additionally, the target inserts one wait state by asserting its ready ($\overline{\text{TRDY}}$) at clock 8.



○ $\overline{\text{DEVSEL}}$ *is sampled by the Am79C974 controller.*

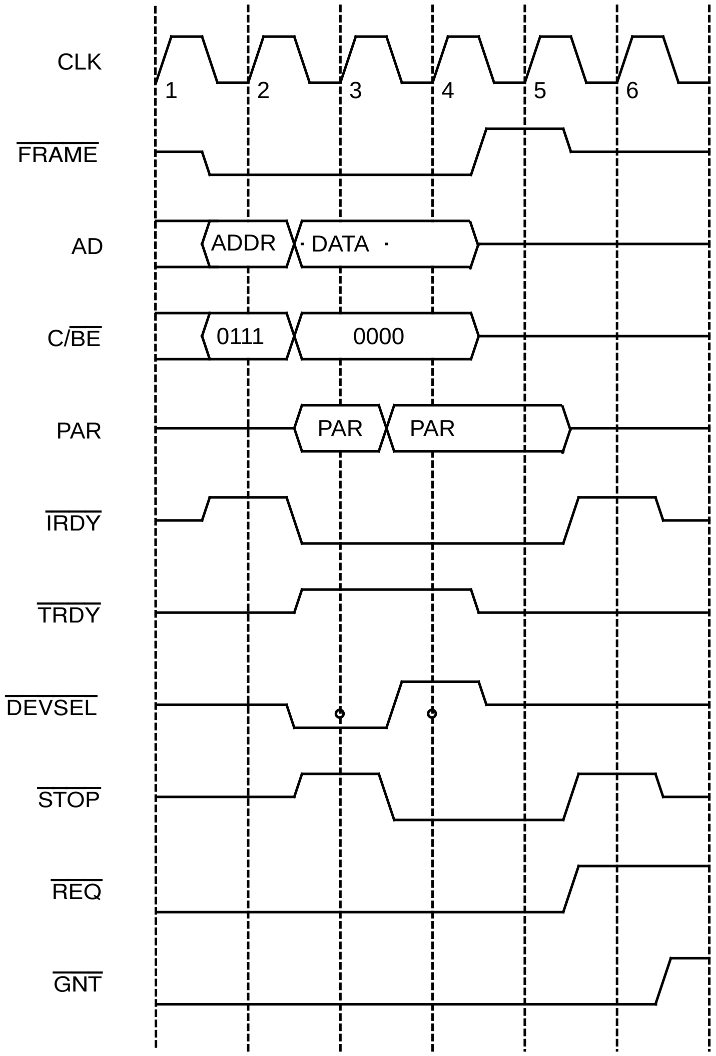
18681A/1-10

**Figure 6. Non-Burst Read Cycles With Wait States**

### Target Abort

Figure 13 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 Am79C974 controller cannot make any assumption about the success of the previous data transfers in the current transaction. The Am79C974 controller terminates the current transfer with the deassertion of FRAME on clock 5 and one clock cycle later with the deassertion of IRDY. It finally releases the bus on clock 6.

Since data integrity is not guaranteed, the Am79C974 controller cannot recover from a target abort event. For Ethernet, the Am79C974 controller will reset all CSR and BCR locations to their H_RESET values. Any on-going network activity will be stopped immediately. The PCI configuration registers will not be cleared. For SCSI, when target aborts, INTA will not be asserted, but the ABORT bit (bit 2 of the DMA status register at offset 54h), is set. For either Ethernet or SCSI a target abort causes RTABORT (bit 12) of the status register in the appropriate PCI configuration space to be set.



○ DEVSEL *is sampled by the Am79C974 controller.*

18681A/1-17

**Figure 13. Target Abort**

### Master Abort

The Am79C974 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 Am79C974 controller. For the Ethernet, the Am79C974 controller will reset all CSR and BCR locations to their H_RESET values. Any on-going network activity will be stopped immediately. The PCI configuration registers will not be cleared. For SCSI, when the master aborts, INTA will not be asserted, but the ABORT bit (bit 2 of the DMA status register at offset 54h), is set. For either Ethernet or SCSI a master abort causes RMABORT (bit 13) of the status register in the appropriate PCI configuration space to be set.

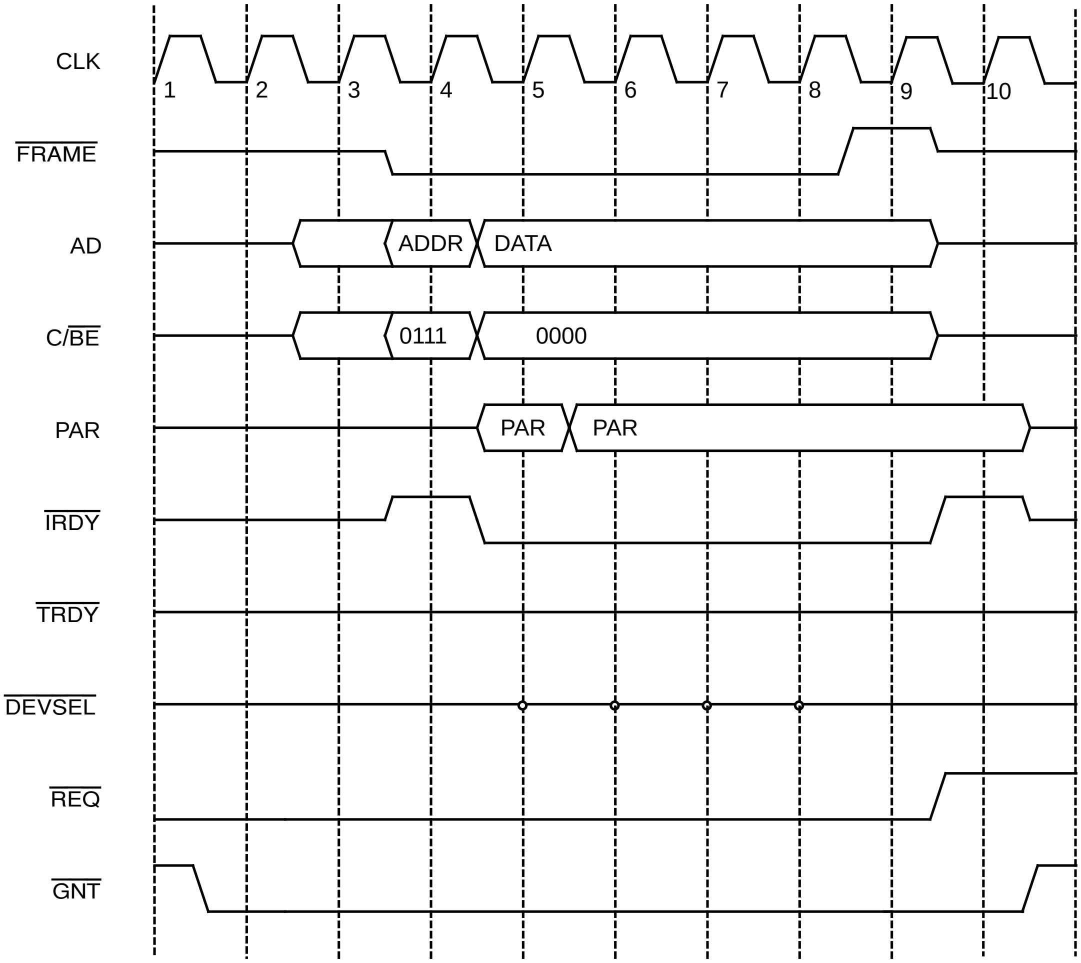


**Figure 16. Master Abort**

Since the PCI bus is 4 bytes wide and the SCSI bus is only 1 byte wide, funneling logic is included in this engine to handle byte alignment and to ensure that data is properly  transferred between the SCSI bus and the wider PCI bus. All boundary conditions are handled through hardware by the DMA Engine.

The DMA engine is also designed for block type (4 KByte page) transfers to support scatter-gather operations. Implementation of this feature is described further in the DMA Scatter-Gather Mechanism section.

### DMA FIFO

Data transfers from the SCSI FIFO to the DMA FIFO take place each time the threshold of  two bytes is reached on the SCSI side. The transfer is initiated by the SCSI block when the internal DREQ is asserted, and continues with the $\overline{\text{DACK}}$ handshaking which typically takes place in DMA accesses. Data is accumulated in the DMA FIFO until a threshold of 16 DWORD (64 bytes) is reached. Data is then burst across the PCI bus to memory. Residue data which is less than the threshold in each FIFO is sent in non-contiguous bursts. For memory read operations, data is sent in burst mode to the DMA FIFO and continues through to the SCSI FIFO and onto the SCSI bus.

### DMA BLAST Command

This command is used to retrieve the contents of the DMA FIFO when the Target disconnects during a DMA Write operation. This could happen for example if a SCSI disk drive detected the end of a sector and decided to give up the bus while it was looking for  the next sector. The Target Disconnect can leave some bytes of data in the DMA FIFO and some in the SCSI FIFO, while some bytes have yet to be transferred from the peripheral device. When this happens, the controller will assert $\overline{\text{INTA}}$ to interrupt the processor, the SCSI state machine will continue to empty its contents into the DMA FIFO, but the DMA FIFO will not necessarily dump its contents into memory (unless the 64-byte DMA threshold happens to have been exceeded at this time).

The BLAST command causes the contents of the DMA FIFO to be emptied into memory. There are some restrictions on when this command should be used.

- First, the command should be used only to recover from an interrupted DMA write operation—not a read operation.
- Second, the command must not be issued until the SCSI FIFO has finished dumping its contents into the DMA FIFO.
- Third, the command should never be issued when the DMA FIFO has already been emptied. This is indicated by the state of the DONE bit in the DMA STATUS register at ((B)+54h).

(This is a test for a special case that can occur when a Target Disconnect leaves only 1 byte left to be transferred from the SCSI peripheral. In this case, if the original transfer count was even, an odd number of bytes will be left in the SCSI FIFO.  Since the SCSI engine transfers data to the DMA FIFO two bytes at a time, the last transfer consists of one byte of valid data and one byte of garbage. The DMA engine treats this final invalid byte as valid data and writes it to memory.  When it does this, it decrements its Working Byte Counter to zero and sets the DONE bit, even though 1 byte still needs to be retrieved from the peripheral device.)

The following procedure outlines the use of the BLAST command after an interrupt has occurred. Note that the order of steps 2–4 is not critical. The order can be changed to tune the performance. Also note that each register is read only once in this procedure even though several tests may be made on data from one register.

1. Verify that INT (bit 7 of the SCSI status register at ((B)+10h) is set to indicate that a SCSI interrupt is pending.
2. Read the SCSI current FIFO count (bits 4:0 of the Current FIFO/Internal State register at ((B)+1Ch). If this value is not zero, wait for the SCSI FIFO to empty its contents into the DMA FIFO.
3. If bit 4 of the SCSI status register (CTZ) is set, stop here. The transfer is complete, and it is not necessary to execute the BLAST command.
4. Verify that DIR (bit 7 of  DMA command register at ((B)+40h) is set to one to indicate that the direction of transfer is from SCSI to memory.
5. Test the error bits in the DMA status register and the SCSI status register (STATREG at ((B)+10h) to verify that the contents of the DMA and SCSI FIFOs are not invalid.
6. Test the DMA DONE bit in the DMA STATUS register. If DONE is not set, write ‘01’ to the DMA command register to issue the BLAST command. This will move the remaining data from the DMA FIFO into memory.
7. Wait until the BLAST complete (BCMPLT) in the DMA STATUS register is set to indicate the completion of the BLAST operation.
8. Write ‘00’ to the DMA command register to issue the IDLE command to the DMA engine. (Note that the IDLE command does not generate an interrupt.)

The above procedure insures that the data that has been transferred from the SCSI peripheral does not get lost in the DMA FIFO when a Target Disconnect occurs. However, it does not complete the original transfer. The software must now read the SCSI Current Transfer Count register (CTCREG) to find out how many bytes have yet to be transferred from the SCSI peripheral device and must start a new transfer operation to get the rest of the data. (CTCREG consists of three bytes located at ((B)+00h, (B)+04h, and (B)+38h.)

### Funneling Logic

Figure 26 shows the internal DMA logic interface with the SCSI block. The DMA FIFO interfaces to the Funnel Logic block via a 32-bit data bus, and the funnel logic properly reduces this stream of data to a 16-bit stream to properly interface with the SCSI FIFO.

18681A/1-30

**Figure 26. DMA FIFO to SCSI FIFO Interface**

### SCSI DMA Programming Sequence

The following section outlines the procedure for executing SCSI DMA operations:

1. Issue IDLE command to the DMA Engine
2. Configure the SCSI block registers (e.g. synchronous operation, offset values, etc.)
3. Program the DMA registers to set up address and transfer count
4. Issue a transfer command to the SCSI command registers
5. Issue the START command to the DMA engine
6. At the end of the DMA transaction, issue the IDLE command to the DMA engine

### MDL Based DMA Programming

The following section outlines the procedure for executing MDL based DMA operation:

1. Set up the MDL list
2. Use the programming sequence defined earlier for initiating a SCSI DMA transfer

## DMA Registers

The following is a summary of the DMA register set or the DMA Channel Context Block (DMA CCB). These registers control the specifics for DMA operations such as transfer length and scatter-gather options. The three read-only working counter registers allow the system software and driver to monitor the DMA transaction. Each register address is represented by the PCI Configuration Base Address (B) and its corresponding offset value. The Base address for the Am79C974 is stored at register address (10h) in the PCI configuration space.

**Table 6. The DMA Registers**

| Register Acronym | Addr (Hex) | Register Description | Type |
|---|---|---|---|
| CMD | (B)+40 | Command (bits 31:8 reserved, bits 7:0 used) | R/W |
| STC | (B)+44 | Starting Transfer Count (bits 31:24 reserved, bits 23:0 used) | R/W |
| SPA | (B)+48 | Starting Physical Address (bits 31:0 used) | R/W |
| WBC | (B)+4C | Working Byte Counter | R |
| WAC | (B)+50 | Working Address Counter (bits 31:0 used) | R |
| STATUS | (B)+54 | Status Register (bits 31:8 reserved, bits 7:0 used) | R |
| SMDLA | (B)+58 | Starting Memory Descriptor List (MDL) Address | R/W |
| WMAC | (B)+5C | Working MDL Counter | R |

## Command Register (CMD)

The upper 3 bytes of Command register are reserved, the remaining (LSB) byte is defined as follows:

**Address (B)+40h, LSB** **READ/WRITE**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DIR | INTE_D | INTE_P | MDL | Reserved | Reserved | CMD1 | CMD0 |

***DIR:***

Data transfer direction bit.

***INTE_D:***

DMA transfer active interrupt bit.

***INTE_P:***

Page transfer active interrupt bit.

***MDL:***

Memory Descriptor List (MDL) SPA enable bit.

***RESERVED:***

Reserved for future expansion. The zero value must be written in these bits.

***CMD1-0:***

These two bits are encoded to represent four commands: IDLE, BLAST, START, and ABORT.

| CMD1 | CMD0 | Command | Description |
|---|---|---|---|
| 0 | 0 | IDLE | Resets the DMA block to the IDLE state. Stops any current transfer. Does not affect status bits or cause an interrupt. |
| 0 | 1 | BLAST | Empties all data bytes in DMA FIFO to memory during a DMA write operation. Upon completion, the 'BCMPLT' bit will be set in the DMA Status register. This command should not be used during a DMA read operation. |
| 1 | 0 | ABORT | Terminates the current DMA transfer. The DMA engine should be restored to the 'IDLE' state following execution of this command.<br>***Note**: This is only valid after a 'START' command is issued.* |
| 1 | 1 | START | Initiates a new DMA transfer. These bits must remain set throughout the DMA operation until the 'DONE' bit in the DMA Status Register is set.<br>***Note:** This command should be issued only after all other control bits have been initialized.* |

### DMA Scatter-Gather Mechanism

The Am79C974 contains a scatter-gather translation mechanism which facilitates faster data transfers. This feature uses a **M**emory **D**escriptor **L**ist (a list of contiguous physical memory addresses) which is stored in system memory. Use of the Memory Descriptor List allows a single SCSI transfer to be read from (or written to) non-contiguous physical memory locations. This mechanism avoids copying the transfer data and MDL list, which was previously required for conventional DMA operations.

#### *Memory Descriptor List (MDL)*

The MDL is a non-terminated (no End Of File marker) list of 32-bit page frame addresses, which is always aligned on a Double Word boundary. The format is shown below:

| 31 … 12 | 11 … 0 |
|---|---|
| Page Frame Address | Ignored |

#### *DMA Scatter-Gather Operation (4k aligned elements)*

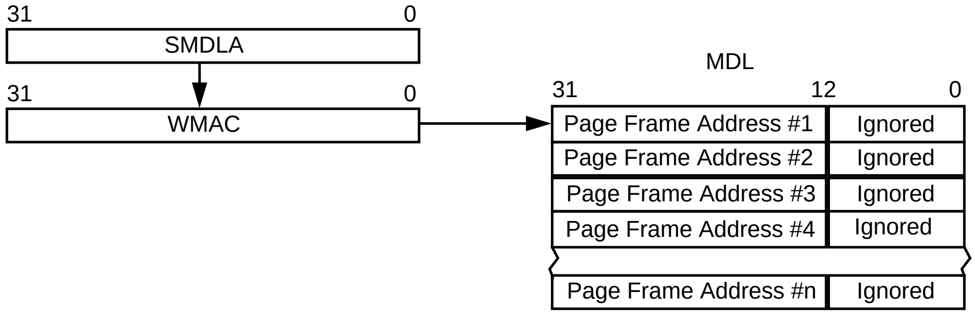
The scatter-gather mechanism described below assumes 4k page alignment and size for all MDL entries except the first and last entry. This feature is enabled by setting the MDL bit in the DMA Command register (Bit 4, Address (B)+40h).

1. a) Prepare the Memory Descriptor List (MDL) through software and store it in system memory.

   b) Load the address of the starting entry in the Memory Descriptor List (MDL) into the Start Memory Descriptor List Address (SMDLA) register. This value is automatically copied into the Working MDL Address Counter (WMAC).

   c) Program the Starting Transfer Count (STC) register with the total transfer length (i.e., # of bytes). Also program the Starting Physical Address (SPA) register (bits 11:0) with the starting offset of the first entry.

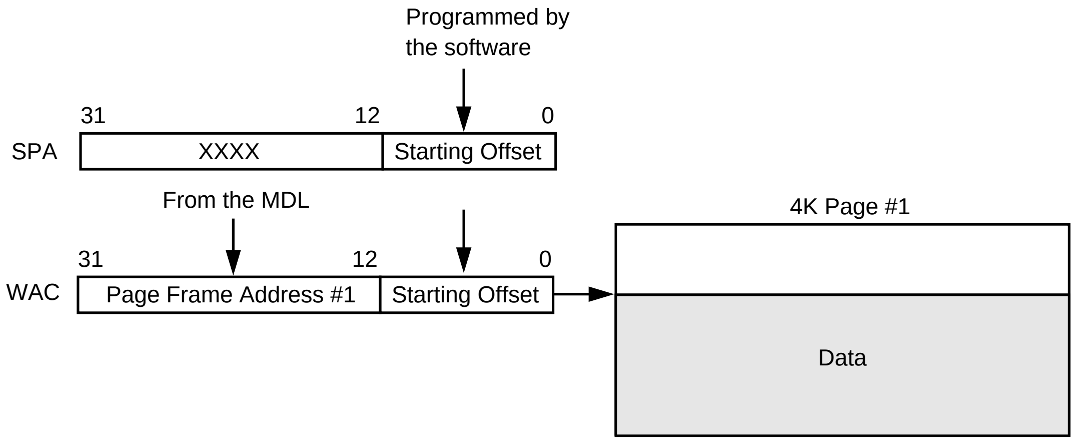
***Note**: The value in the SMDLA register must be double word aligned. Therefore, read/write transactions will always begin on a double word boundary.*



18681A/1-31

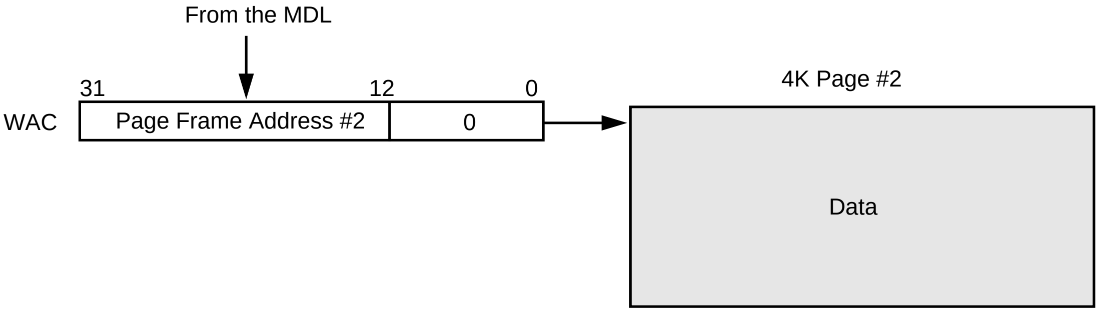
In this example, the contents of the WMAC register is pointing to page frame address #1. When the first entry in the MDL in read (page frame address #1), the WMAC register is incremented to point to the next page entry (page frame address #2).

2. Issue the Start DMA Command. The Am79C974 reads the page frame address (bits 31:12) from the MDL entry and combines it with the first page offset value in the Starting Physical Address (SPA) register (bits 11:0). This 32-bit value is loaded into the Working Address Counter (WAC) register and becomes the physical address for page#1, as shown below. The WMAC is then incremented to point to the next entry in the MDL.



18681A/1-32

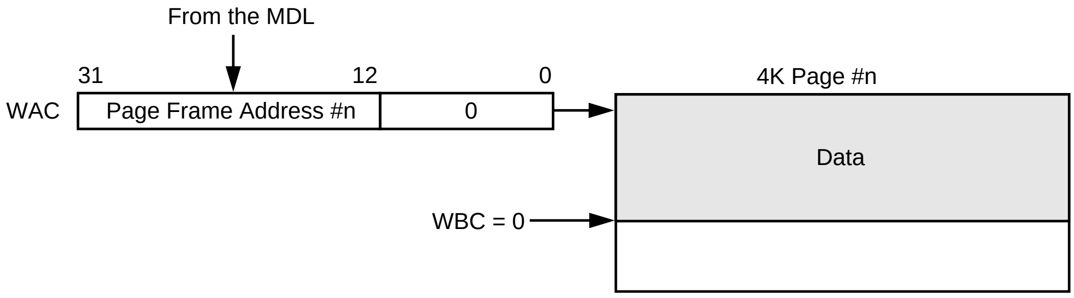
3. When the WAC register (bits 11:0) reaches the first 4K byte boundary, the Am79C974 reads the second MDL entry and combines the page frame address (bits 31:12) from this entry of the MDL with bits 11:00 of the WAC register. This becomes the physical address for page#2. Since the WAC register (bits 11:0) has rolled over to ‘00h’, the WAC now points to the beginning of the Page Frame Address #2 as shown below. The WMAC is then incremented to point to the next entry in the MDL.



18681A/1-33

When WAC (bits 11:0) again reaches the next 4K byte boundary, the next MDL entry is read into the WAC. The operation continues in this way until WMAC register reaches the last MDL entry (Page Frame Address #n in this example).

4. The WAC register points to the beginning of the last page#n and the DMA operation continues until the byte count is exhausted in the Working Byte Counter (WBC) register. When WBC=0, the chip stops incrementing the WAC register. This is shown below.



18681A/1-34

***DMA Scatter-Gather Operation (Non-4k aligned elements MDL not set)***

There is another way to implement a scatter-gather operation which does not force the data elements to be aligned on 4k boundaries. It assumes a “traditional” scatter-gather list of the following format:

| | |
|---|---|
| Element 0 | Physical Address Byte Count |
| Element 1 | Physical Address Byte Count |
| | ... |
| Element n | Physical Address Byte Count |

This second implementation is described as follows:

1. Set the SCSI Start Transfer Count Register ((B)+00h, (B)+04h, (B)+38h) to the Byte count of the first Scatter-Gather element.
2. Program the DMA Starting Transfer Count Register ((B)+44h) to the Byte Count of the first Scatter-Gather element.
3. Program the DMA Starting Physical Address Register ((B)+48h) to the Physical Address of the first Scatter-Gather element.
4. Start the SCSI operation by issuing a SCSI Information Transfer command.
5. Start the DMA Engine with DMA Transfer Interrupt Enable (Bit 6, (B)+40h).
6. When the Scatter-Gather element’s Byte Count is exhausted, the DMA engine will generate an interrupt.
7. Reprogram the next Scatter-Gather element’s Byte Count into the SCSI Start Transfer Count Register and the DMA Starting Transfer Count Register.
8. Reprogram the DMA Starting Physical Address Register ((B)+48h) to the Physical Address of the next Scatter-Gather element.
9. Repeat steps 4–8 until the Scatter-Gather list is completed.

18681A/1-67

*where C1 – C9 are decoupling capacitors.*

**Figure E-5. Decoupling Capacitor Placement**