# DECchip 21041 PCI Ethernet LAN Controller

# Hardware Reference Manual

Order Number: EC–QAWXA–TE

**Revision/Update Information:** This is a preliminary manual.

**Digital Equipment Corporation**
**Maynard, Massachusetts**

**April 1995**



This document was prepared using VAX DOCUMENT Version 2.1.

# Contents

## 4 Host Communication Area

## 5 Functional Description



## 6 Host Bus Operation

## 7 Network Interface

## D Technical Support, Ordering, and Associated Literature



## Index

## Examples



## Figures

## Tables

# Preface

## Purpose and Audience

This manual describes the operation of the DECchip 21041 PCI Ethernet LAN Controller (also referred to as the 21041).

The 21041 operates in two modes: mode 0 and mode 1. In mode 0, the 21041 is software-compatible with the DECchip 21040 and functions as a DECchip 21040. In mode 1, the 21041 incorporates the enhanced features listed in Chapter 1. For details about mode 0 operation, refer to the *DECchip 21040 Ethernet LAN Controller for PCI Data Sheet* and the *DECchip 21040 Ethernet LAN Controller for PCI Hardware Reference Manual*.

This manual is intended for system designers who use the 21041 in mode 1.

## Manual Organization

This manual contains eight chapters, four appendixes, and an index.

- Chapter 1, Introduction, includes a general description of the 21041. It also provides an overview of the 21041 hardware components.
- Chapter 2, Signal Descriptions and Bus Commands, provides the physical layout of the 21041 and describes the input and output signals.
- Chapter 3, Registers, provides a complete bit description of the 21041 command and status registers as well as the configuration registers.
- Chapter 4, Host Communication Area, describes how the 21041 communicates with the host using descriptor lists and data buffers.
- Chapter 5, Functional Description, describes reset commands, interrupt handling, startup, and the transmit and receive processes.
- Chapter 6, Host Bus Operation, provides a description of the read, write, and termination cycles.

- Chapter 7, Network Interface, describes the 10BASE-T and AUI interfaces. It includes a complete description of media access control operations and provides detailed information about the transmit and receive functions.
- Chapter 8, Boot ROM and Serial ROM Interfaces, describes the boot ROM and MICROWIRE serial ROM interfaces, connection, and read and write cycles.
- Appendix A, Joint Test Action Group Test Logic, provides descriptions of the testing, observing, and modifying circuit activity during normal operation.
- Appendix B, DNA CSMA/CD Counters and Events Support, describes features that support the driver in implementing and reporting the specified counters and events.
- Appendix C, Hash C Routine, provides two examples of C routines that generate a hash index for a given Ethernet address.
- Appendix D, Technical Support, Ordering, and Associated Literature, contains information about technical support and ordering Digital Semiconductor products and documentation.
- The index provides an alphabetical list of topics described in this manual. An entry with an f appended to the page number (for example, 21041 pinout diagram, 2-2f) indicates a figure reference. An entry with a t appended to the page number (for example, Twisted-pair compensation behavior, 3-60t) indicates a table reference.

## Document Conventions

The values 1, 0, and X are used in some tables. X signifies a don't care (1 or 0) condition which can be determined by the system designer.

# 1
# Introduction

This chapter provides a general description of the DECchip 21041 PCI Ethernet LAN Controller (21041). It describes the 21041 features and provides an overview of the hardware.

## 1.1 General Description

The 21041 is an Ethernet LAN controller that is based on the peripheral component interconnect (PCI) local bus. The 21041 provides a glueless connection to the PCI and a direct interface to a boot ROM of up to 256K-bytes. It supports PCI 5.0-volt and 3.3-volt signaling environments and a power-down mode for energy conservation.

During host interface operation, the 21041 interfaces with the processor using onchip command and status registers (CSRs) and a shared host memory area that is set up mainly during initialization. This minimizes processor involvement in the 21041 operation during normal reception and transmission. Bus traffic is also minimized by filtering out received runt frames and by automatically retransmitting collided frames without a repeated fetch from host memory.

During communication interface operation, the 21041 provides both an attachment unit interface (AUI) and a twisted-pair (TP) interface, enabling a low chip count connection to the two most popular Ethernet interfaces. The 21041 can sustain transmission or reception of minimal-sized, back-to-back packets at full line speed with a 9.6-microsecond interpacket gap. The 21041 also supports full-duplex operations and autodetection of full- and half-duplex network environments using the IEEE 802.3 100BASE-T autonegotiation algorithm.

## 1.2 Features

The 21041 has the following features:

- Offers a single-chip Ethernet controller for the PCI local bus
  - Provides a glueless connection to the PCI bus
  - Contains an onchip integrated attachment unit interface (AUI) port and a 10BASE-T transceiver
  - Implements pinout order as recommended by the *PCI Local Bus Specification* to allow board layout within trace-length restriction
  - Supports promiscuous Ethernet reception
- Implements the same architecture as the DECchip 21040 to allow using unified drivers
- Provides an upgradable boot ROM interface (flash or EEPROM) of 64K bytes, 128K bytes, or 256K bytes
- Contains serial ROM interface (suitable also for MICROWIRE EEPROM) for Ethernet ID address and other system parameters (optional)
- Supports full-duplex operation and IEEE 802.3 autonegotiation algorithm of full- and half-duplex environments
- Provides clock speed of up to 33 megahertz, with no wait states on PCI master operation
- Enables powerful onchip DMA with programmable burst sizes of up to 32 longwords providing low CPU utilization
- Optimizes full-duplex operation by implementing a unique, patent-pending, intelligent arbitration between DMA channels to prevent underflow or overflow
- Incorporates a 16-bit, general-purpose timer
- Contains two large (256-byte) independent receive and transmit FIFOs
- Supports big or little endian byte ordering for buffers and descriptors
- Implements joint test action group (JTAG) compatible test access port with boundary-scan pins
- Provides full support of IEEE 802.3, ANSI 8802-3, and Ethernet standards
- Offers a unique, patented solution to Ethernet capture-effect problem

- Contains a variety of flexible address filtering modes
  - 16 perfect addresses
  - 512 hash-filtered multicast addresses and one perfect address
  - 512 hash-filtered physical addresses and multicast addresses
  - Inverse perfect filtering
- Supports seven LEDs including receive, receive address match, transmit, transmit jabber, collision, linkpass, and polarity
- Enables automatic detection and correction of 10BASE-T receive polarity
- Enables full autosensing between the 10BASE-T, 10BASE2, and 10BASE5 ports
- Provides external and internal loopback capability
- Provides a software-controllable, power-saving mode
- Uses a 3.3-volt CMOS device that interfaces to 5.0- or 3.3-volt logic
- Supports both PCI 5.0-volt and 3.3-volt signaling environments

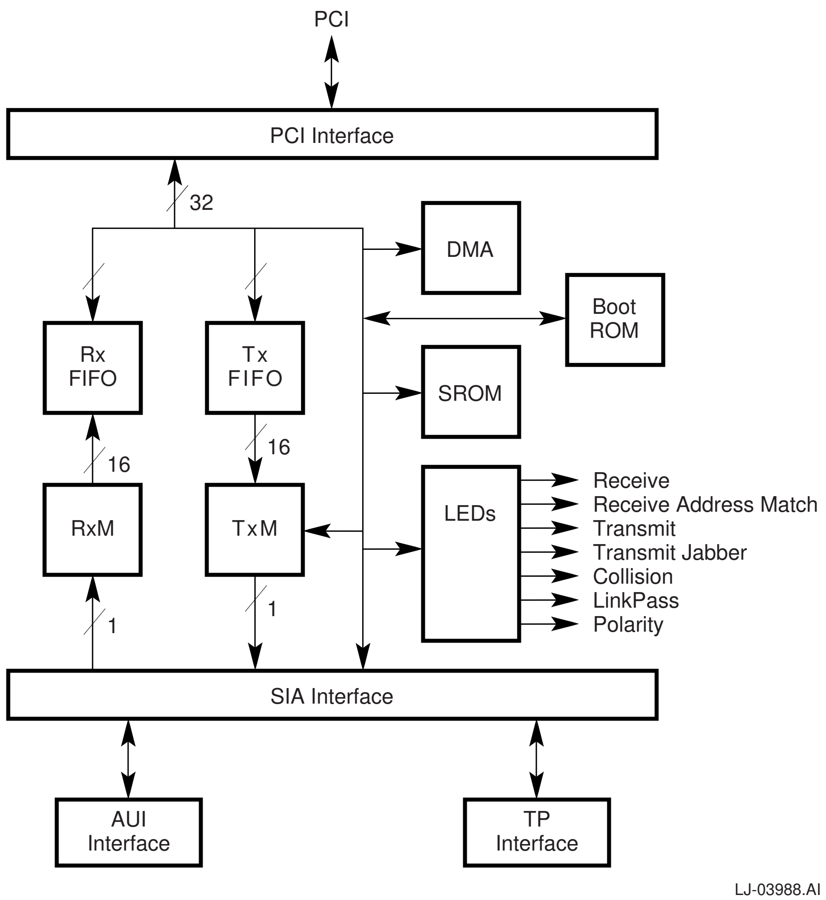
## 1.3 Hardware Overview

The following list describes the 21041 hardware components, and Figure 1–1 shows a block diagram of the 21041.

- PCI Interface—Includes all interface functions to the PCI bus; handles all interconnect control signals, and executes PCI direct memory access (DMA) and I/O transactions.
- DMA—Contains dual receive and transmit controller; supports bursts of up to 32 longwords; handles data transfers between system memory and onchip memory.
- FIFOs—Contains dual 256-byte FIFOs for receive and transmit; supports automatic packet deletion (runt packets or after a collision) and packet re-transmission after a collision on transmit.
- TxM—Handles all CSMA/CD[1] MAC[2] transmit operations and transfers data from transmit FIFO to the serial interface attachment (SIA) for transmission.
- RxM—Handles all CSMA/CD receive operations and transfers the data from the SIA to the receive FIFO.
- SIA Interface—Performs physical layer operations; implements the AUI and 10BASE-T functions including the Manchester encoder and decoder functions.
- Boot ROM—Includes the required interface to the boot ROM for read and write operations. Accesses to bytes or to longwords (32-bit) are supported.
- SROM—Provides all interface logic to allow accesses to the external serial ROM.
- LEDs—Contain the logic to detect seven network events and drive them outside the device for LED display. Also supports two general-purpose LEDs.

[1] Carrier-sense multiple access with collision detection
[2] Medium access control

**Figure 1–1 DECchip 21041 Block Diagram**



# 2

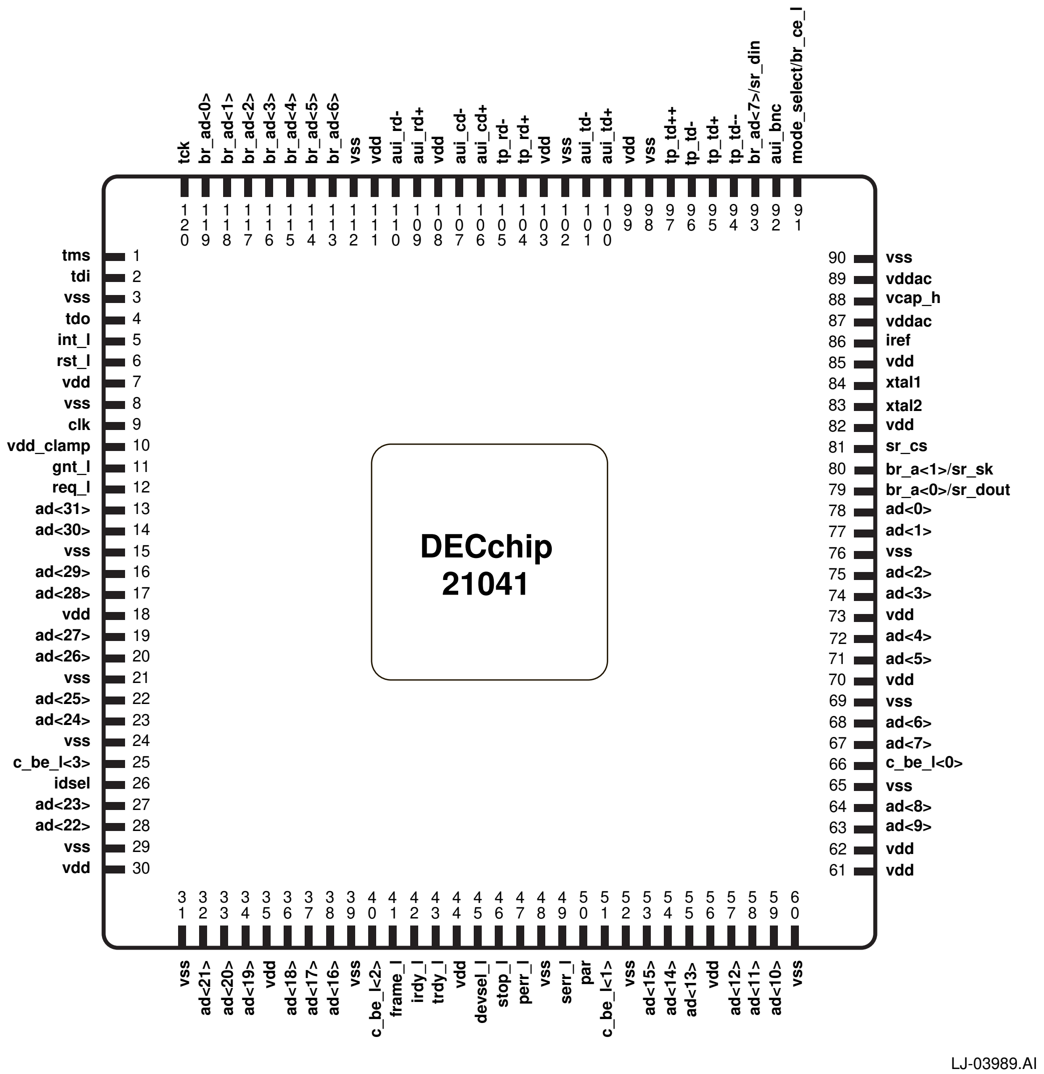
# Signal Descriptions and Bus Commands

This chapter provides the DECchip 21041 PCI Ethernet LAN Controller (21041) pinout, describes the 21041 signals, and lists the bus commands.

## 2.1 21041 Pinout

The 21041 is housed in the 120-pin plastic quad flat pack. The 21041 uses all pins. Figure 2–1 shows the 21041 pinout.

Figure 2–1 DECchip 21041 Pinout Diagram (Top View)



## 2.2 Signal Descriptions

Table 2–1 describes the signals used by the 21041. The signals are listed alphabetically.

The following terms describe the 21041 pinout.

- Address phase

  Address and appropriate bus commands are driven during this cycle.

- Data phase

  Data and the appropriate byte enable codes are driven during this cycle.

- _l

  All pin names with the _l suffix are only asserted low.

---

**Note**

The following abbreviations are used in the tables in this section.

I = Input
O = Output
I/O = Input/output
O/D = Open drain

---

**Table 2–1 Signal Pin Reference**

| Signal | Type | Description |
|---|---|---|
| ad<31:00> | I/O | 32-bit multiplexed PCI address and data lines. Address and data bits are multiplexed on the same pins. During the first clock cycle of a transaction, ad<31:00> contains a physical byte address (32 bits). During subsequent clock cycles, ad<31:00> contains data. A 21041 bus transaction consists of an address phase followed by one or more data phases. The 21041 supports both read and write bursts. Little and big endian byte ordering can be used. |
| aui_cd– | I | Attachment unit interface receive collision differential negative data. |
| aui_cd+ | I | Attachment unit interface receive collision differential positive data. |
| aui_rd– | I | Attachment unit interface receive differential negative data. |
| aui_rd+ | I | Attachment unit interface receive differential positive data. |
| aui_td– | O | Attachment unit interface transmit differential negative data. |
| aui_td+ | O | Attachment unit interface transmit differential positive data. |
| aui_bnc | I/O | Attachment unit interface (AUI) and BNC select line. When asserted high, the 10BASE5 (AUI) mode is selected. When asserted low, the 10BASE2 (BNC) mode is selected. This output pin is mainly used to enable the external BNC transceiver in 10BASE2 mode. The AUI_BNC pin is an input in mode 0. |

(continued on next page)

**Table 2–1 (Cont.) Signal Pin Reference**

| Signal | Type | Description |
|---|---|---|
| br_ad<7>/sr_din | I/O | Boot ROM address and data line 7. This multiplexed pin carries, in two consecutive address cycles, the boot ROM address bits 7 and 15. During the data phase, it contains data.<br>When the 21041 operates with the serial ROM, serial ROM data in (sr_din) serially shifts the written data from the 21041 into the serial ROM device.<br>During operation with the external register, this pin is used to carry data bit 7 to and from the external register.<br>When mode 0 (DECchip 21040) is selected, this pin must be tied to VSS. |
| br_ad<6:0> | I/O | Boot ROM address and data multiplexed lines bits 6 through 0. In two consecutive address cycles, these lines contain the boot ROM address bits 6 through 2, oe_l, and we_l in the first cycle, and these lines contain boot ROM address bits 14 through 8 in the second cycle. During the data cycles, bits 6 through 0 contain data.<br>br_ad<6:0> can be connected directly to the following seven network event LEDs: Transmit, Collision, Receive Address Match, Receive, Polarity, Transmit Jabber, and LinkPass respectively.<br>During operation with the external register, these lines are used to carry data bits 6 through 0 from and to the external register. |
| br_a<1>/sr_sk | O | Boot ROM address line bit 1. This multiplexed pin is used to latch the boot ROM address and control lines by the two external latches.<br>When the 21041 operates with the serial ROM, sr_sk provides the serial ROM clock. |

(continued on next page)

**Table 2–1 (Cont.) Signal Pin Reference**

| Signal | Type | Description |
|---|---|---|
| br_a<0>/sr_dout | I/O | Boot ROM address line bit 0. This multiplexed pin carries the boot ROM address 0. In a 256K-byte configuration, this pin also carries, in two consecutive address cycles, boot ROM address bits 16 and 17.<br><br>When the 21041 operates with the serial ROM, serial ROM data out (sr_dout) serially shifts the read data from the serial ROM device into the 21041.<br><br>During operation with the external register, this pin is used for read and write control. |
| c_be_l<03:00> | I/O | Bits 0 through 3 of the bus command and byte enable lines. Bus command and byte enable are multiplexed on the same PCI pins. Table 2–2 lists the bus commands.<br><br>During the address phase of the transaction, c_be_l<03:00> provide the bus command.<br><br>During the data phase, c_be_l<03:00> provide the byte enable. The byte enable determines which byte lines carry valid data. For example, c_be_l<00> applies to byte 0, and c_be_l<03> applies to byte 3.<br><br>In all master and I/O operations, c_be_l<03:00> contain a value equal to a longword hexadecimal value of 0. In configuration operations, c_be_l<03:00> can contain any value; the 21041 supports byte, word, and longword operations. |
| clk | I | The clock provides the timing for the 21041–related bus transactions. All other bus signals are sampled on the rising edge of clk. The clock range is between 16 and 33 megahertz. |
| devsel_l | I/O | Device select is asserted when it is the target of the current bus access. When the 21041 is the initiator of the current bus access, it expects the target to assert devsel_l within five bus cycles, confirming the access. To accomplish this, the 21041 asserts this signal in a medium speed (within two bus cycles). If the target does not assert devsel_l within the required bus cycles, the 21041 aborts the cycle. |

(continued on next page)

**Table 2–1 (Cont.) Signal Pin Reference**

| Signal | Type | Description |
|---|---|---|
| frame_l | I/O | Cycle frame is driven by the 21041 (bus master) to indicate the beginning and duration of an access. frame_l asserts to indicate the beginning of a bus transaction. While frame_l is asserted, data transfers continue. frame_l deasserts to indicate that the next data phase is the final data phase transaction. |
| gnt_l | I | Bus grant asserts to indicate to the 21041 that access to the bus is granted. |
| idsel | I | Initialization device select asserts to act as a chip select during configuration read or write transactions. |
| int_l | O/D | Interrupt request asserts when one of the appropriate bits of CSR5 sets and causes an interrupt, provided that the corresponding mask bit in CSR7 is not asserted. int_l deasserts by writing a 1 into the appropriate CSR5 bit.<br><br>If more than one interrupt bit is asserted in CSR5, the host clears only the interrupt bit that was acknowledged, int_l deasserts for one cycle and then asserts again. This process continues until all interrupts are acknowledged.<br><br>When deasserted, this pin should be pulled up by an external resistor. |
| irdy_l | I/O | Initiator ready indicates the bus master's ability to complete the current data phase of the transaction.<br><br>A data phase is completed on any clock when both irdy_l and target ready (trdy_l) are asserted. Wait cycles are inserted until both irdy_l and trdy_l are asserted together.<br><br>When the 21041 is the bus master, irdy_l is asserted during write operations to indicate that valid data is present on ad<31:00>. During read operations, the 21041 asserts irdy_l to indicate that it is ready to accept data. |
| iref | I | Current reference input for the analog phase-locked loop (PLL) logic. |

(continued on next page)

**Table 2–1 (Cont.) Signal Pin Reference**

| Signal | Type | Description |
|---|---|---|
| mode_select/br_ce_l | I/O | Selects either mode 0 (DECchip 21040) or mode 1 (DECchip 21041) operation. When this pin is tied to VSS, mode 0 operation is selected. When this pin is not connected (internally pulled up), mode 1 operation is selected. When a hardware or software reset is detected, the 21041 senses this input and sets the mode of operation accordingly. |
| | | When mode 1 operation is selected, this pin changes its functionality, and it is actually the boot ROM or the external register chip enable (br_ce_l) output pin. |
| par | I/O | Parity is calculated by the 21041 as an even parity bit for the ad<31:00> and c_be_l<03:00> lines. |
| | | During address and data phases, par is calculated on all the ad and c_be_l lines whether or not any of these lines carry meaningful information. |
| perr_l | I/O | Parity error asserts when a data parity error is detected. |
| | | When the 21041 is the bus master and a parity error is detected, the 21041 asserts both CSR5 bit 13 (system error) and CFCS bit 8 (serr_l enable) and completes the current data burst transaction, then stops its operation. After the host clears the system error, the 21041 continues its operation. |
| | | When the 21041 is the bus target and a parity error is detected, the 21041 asserts perr_l. |
| req_l | O | Bus request is asserted by the 21041 to indicate to the bus arbiter that it wants to use the bus. |
| rst_l | I | Resets the 21041 to its initial state. This signal must be asserted for at least 10 active PCI clock cycles. When in the reset state, all PCI port output pins are put into tristate and all open drain signals are floated. |

(continued on next page)

**Table 2–1 (Cont.) Signal Pin Reference**

| Signal | Type | Description |
|---|---|---|
| serr_l | O/D | If an address parity error is detected and CFCS bit 31 (detected parity error) is enabled, the 21041 asserts both serr_l (system error) and CFCS bit 30 (signal system error).<br>When an address parity error is detected, system error asserts two clocks after the failing address.<br>When deasserted, this pin should be pulled up by an external resistor. |
| sr_cs | I/O | Serial ROM chip select asserts when the 21041 accesses the serial ROM. |
| stop_l | I/O | Stop indicator indicates that the current target is requesting the bus master to stop the current transaction.<br>The 21041 responds to the assertion of stop_l when it is the bus master, either to disconnect, retry, or abort. |
| tck | I | JTAG clock shifts state information and test data into and out of the 21041 during JTAG test operations (Appendix A). |
| tdi | I | JTAG data in is used to serially shift test data and instructions into the 21041 during JTAG test operations (Appendix A). |
| tdo | O | JTAG data out is used to serially shift test data and instructions out of the 21041 during JTAG test operations (Appendix A). |
| tms | I | JTAG test mode select controls the state operation of JTAG testing in the 21041 (Appendix A). |
| tp_rd– | I | Twisted-pair negative differential receive data from the twisted-pair lines. |
| tp_rd+ | I | Twisted-pair positive differential receive data from the twisted-pair lines. |
| tp_td–<br>tp_td– – | O | Twisted-pair negative differential transmit data. The positive and negative differential transmit data outputs are resistively combined outside the 21041 with equalization to compensate for intersymbol interference on the twisted-pair medium. |

(continued on next page)

**Table 2–1 (Cont.) Signal Pin Reference**

| Signal | Type | Description |
|---|---|---|
| tp_td+<br>tp_td+ + | O | Twisted-pair positive differential transmit data. The positive and negative differential transmit data outputs are resistively combined outside the 21041 with equalization to compensate for intersymbol interference on the twisted-pair medium. |
| trdy_l | I/O | Target ready indicates the target agent's ability to complete the current data phase of the transaction.<br>A data phase is completed on any clock when both trdy_l and initiator ready (irdy_l) are asserted. Wait cycles are inserted until both irdy_l and trdy_l are asserted together.<br>When the 21041 is the bus master, trdy_l is asserted by the bus slave on the read operation indicating that valid data is present on ad<31:00>. During a write cycle, it indicates that the target is prepared to accept data. |
| vcap_h | I | Capacitor input for analog phase-locked loop logic. |
| vdd | I | 3.3-volt supply input voltage. |
| vddac | I | 3.3-volt supply input for analog phase-locked loop logic. |
| vdd_clamp | I | Supplies +5-volt or +3.3-volt reference for the clamp logic. |
| vss | – | Ground pin. |
| xtal1 | I | Crystal oscillator input. |
| xtal2 | O | Crystal feedback output pin used for crystal connections only. If this pin is unused, do not connect it. |

## 2.3 Bus Commands

Table 2–2 lists the bus commands.

**Table 2–2 Bus Commands**

| c_be_l<3:0> | Command | Type of Support |
|---|---|---|
| 0000 | Interrupt acknowledge | Not supported |
| 0001 | Special cycle | Not supported |
| 0010 | I/O read | Supported as target |
| 0011 | I/O write | Supported as target |
| 0100 | Reserved | – |
| 0101 | Reserved | – |
| 0110 | Memory read | Supported as initiator and target |
| 0111 | Memory write | Supported as initiator and target |
| 1000 | Reserved | – |
| 1001 | Reserved | – |
| 1010 | Configuration read | Supported as target |
| 1011 | Configuration write | Supported as target |
| 1100 | Reserved | – |
| 1101 | Memory write and invalidate | Not supported |
| 1110 | Memory read long | Not supported |
| 1111 | Postable memory write | Not supported |

# 3 Registers

This chapter describes the DECchip 21041 PCI Ethernet LAN Controller (21041) configuration registers as well as the command and status registers (CSRs). The 21041 uses nine configuration registers for initialization and configuration. The configuration registers identify and query the 21041.

The 21041 contains 12 CSRs (CSR0 through CSR11) for communication with the driver to the host. It communicates with the serial interface attachment (SIA) using four additional command and status registers (CSR12 through CSR15).

CSRs are located in the 21041 and are mapped in the host I/O or memory address space. CSRs are used for initialization, pointers, commands, and status reporting.

## 3.1 21041 Configuration Operation

The 21041 enables a full software-driven initialization and configuration. This permits the software to identify and query the 21041.

The 21041 treats configuration space write operations to registers that are reserved as no-ops. That is, the access completes normally on the bus and the data is discarded. Read accesses, to reserved or non-implemented registers, complete normally and a data value of 0 is returned.

Software reset (CSR0<0>) has no effect on the configuration registers. Hardware reset clears the configuration registers.

The 21041 supports byte, word, and longword accesses to configuration address space.

### 3.1.1 Configuration Register Mapping

Table 3–1 lists the definitions and addresses for the configuration registers.

**Table 3–1 Configuration Register Mapping**

| Configuration Register | Identifier | I/O Address |
|---|---|---|
| Identification | CFID | 00H |
| Command and status | CFCS | 04H |
| Revision | CFRV | 08H |
| Latency timer | CFLT | 0CH |
| Base I/O address offset | CBIO | 10H |
| Base memory address | CBMA | 14H |
| Reserved | – | 18H—2CH |
| Expansion ROM base address | CBER | 30H |
| Reserved | – | 34H—38H |
| Interrupt | CFIT | 3CH |
| Driver area | CFDA | 40H |

## 3.2 Configuration Registers

The 21041 implements nine configuration registers. The following subsections describe these registers.

### 3.2.1 Configuration ID Register (CFID)

The CFID register identifies the 21041. Figure 3–1 shows the CFID register bit fields, and Table 3–2 describes the bit fields.

**Figure 3–1 CFID Configuration ID Register**

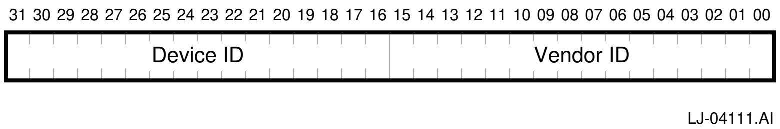


**Table 3–2 CFID Configuration ID Register Description**

| Field | Description |
|---|---|
| **31:16** | **Device ID** |
| | Provides the unique 21041 ID number (0014H). |
| **15:0** | **Vendor ID** |
| | Specifies the manufacturer of the 21041 (1011H). |

Table 3–3 lists the access rules for the CFID register.

**Table 3–3 CFID Access Rules**

| Category | Description |
|---|---|
| Value after hardware reset | 00141011H |
| Read access rules | – |
| Write access rules | Writing has no effect. |

### 3.2.2 Command and Status Configuration Register (CFCS)

The CFCS register is divided into two sections: a command register CFCS<15:0> and a status register CFCS<31:16>.

The command register provides control of the 21041's ability to generate and respond to PCI cycles. When 0 is written to this register, the 21041 logically disconnects from the PCI bus for all accesses except configuration accesses.

The status register records status information for the PCI bus-related events. The CFCS status bits do not clear when read. Writing 1 to these bits clears them; writing 0 has no effect.

Figure 3–2 shows the CFCS bit fields, and Table 3–4 describes the bit fields.

**Figure 3–2 CFCS Command and Status Configuration Register**

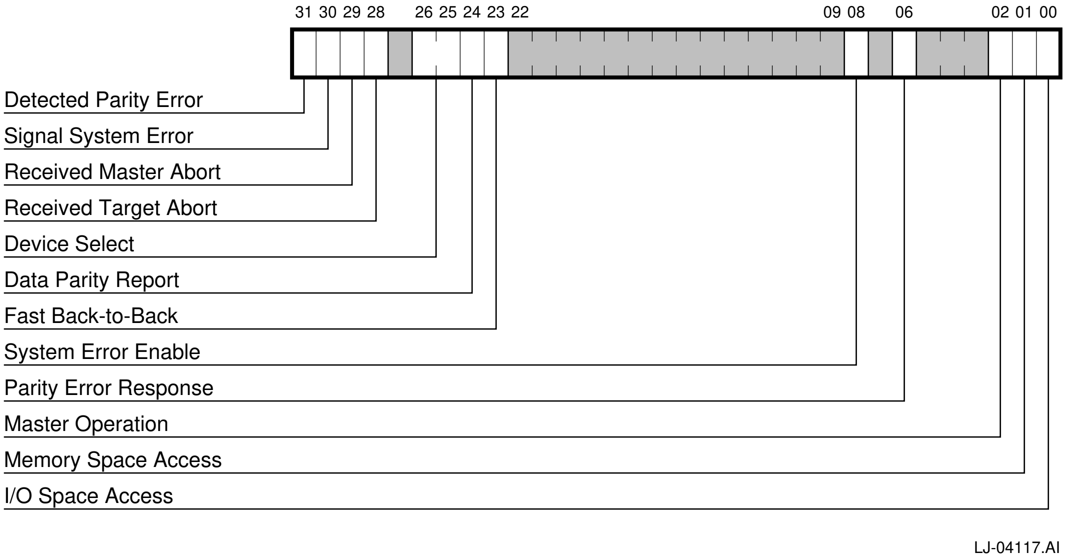


**Table 3–4 CFCS Command and Status Configuration Register Description**

| Field | Description |
|---|---|
| | **Status Registers** |
| **31** | **Detected Parity Error**<br>When set, indicates that the 21041 detected a parity error even if parity error handling is disabled in parity error response CFCS<6>. |
| **30** | **Signal System Error**<br>When set, indicates that the 21041 asserted the system error (serr_l) pin. |
| **29** | **Received Master Abort**<br>When set, indicates that the 21041 terminated a master transaction with master abort. |
| **28** | **Received Target Abort**<br>When set, indicates that the 21041 master transaction was terminated due to target abort. |
| **26:25** | **Device Select**<br>Indicates the timing of the assertion of device select (devsel_l). These bits are fixed at 01 which indicates a medium assertion of devsel_l. |
| **24** | **Data Parity Report**<br>This bit sets when all the following conditions are met:<br>• 21041 asserts parity error (perr_l) or it senses the assertion of perr_l by another device.<br>• 21041 operates as a bus master for the operation that caused the error.<br>• Parity error response CFCS<6> is set. |
| **23** | **Fast Back-to-Back**<br>Always set by the 21041. This indicates that the 21041 is capable of accepting fast back-to-back transactions. |

(continued on next page)

**Table 3–4 (Cont.) CFCS Command and Status Configuration Register Description**

| Field | Description |
|---|---|
| | **Command Registers** |
| **8** | **System Error Enable**<br>When set, the 21041 asserts system error (serr_l) when it detects a parity error on the address phase (ad<31:00> and c_be_l<03:00>). |
| **6** | **Parity Error Response**<br>When set, the 21041 asserts system error CSR5<13> after a parity error detection.<br>When reset, any detected parity error is ignored and the 21041 continues normal operation.<br>Parity checking is disabled after reset. |
| **2** | **Master Operation**<br>When set, the 21041 is capable of acting as a bus master.<br>When reset, the 21041 capability to generate PCI accesses is disabled.<br>For normal 21041 operation, this bit must be set. |
| **1** | **Memory Space Access**<br>When set, the 21041 responds to memory space accesses.<br>When reset, the 21041 does not respond to memory space accesses. |
| **0** | **I/O Space Access**<br>When set, the 21041 responds to I/O space accesses.<br>When reset, the 21041 does not respond to I/O space accesses. |

Table 3–5 lists the access rules for the CFCS register.

**Table 3–5 CFCS Access Rules**

| Category | Description |
|---|---|
| Value after hardware reset | 02800000H |
| Read access rules | – |
| Write access rules | Written during configuration cycle. |

### 3.2.3 Configuration Revision Register (CFRV)

The CFRV register contains the 21041 revision number. Figure 3–3 shows the CFRV bit fields, and Table 3–6 describes the bit fields.

**Figure 3–3 CFRV Configuration Revision Register**

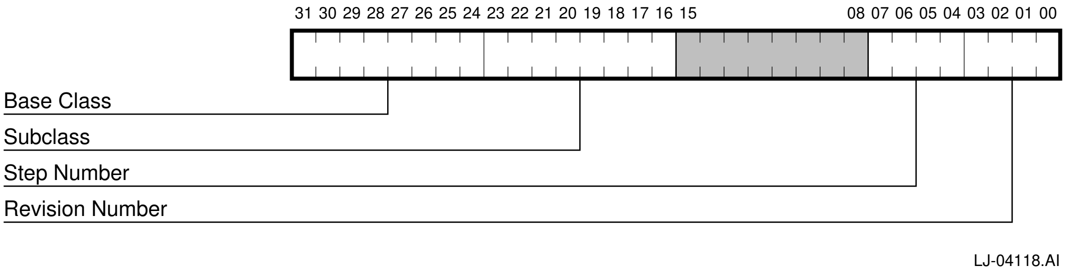



**Table 3–6 CFRV Configuration Revision Register Description**

| Field | Description |
|---|---|
| **31:24** | **Base Class** |
| | Indicates the network controller and is equal to 2H. |
| **23:16** | **Subclass** |
| | Indicates the Ethernet controller and is equal to 0H. |
| **7:4** | **Step Number** |
| | Indicates the 21041 step number and is equal to 1H. This number is incremented for subsequent 21041 steps. |
| **3:0** | **Revision Number** |
| | Indicates the 21041 revision number. This number is incremented for subsequent 21041 revisions within the current step. |

Table 3–7 lists the access rules for the CFRV register.

**Table 3–7 CFRV Access Rules**

| Category | Description |
|---|---|
| Value after hardware reset | 0200001*X*H where *X* is revision number. |
| Read access rules | – |
| Write access rules | Writing has no effect. |

### 3.2.4 Configuration Latency Timer Register (CFLT)

This register configures the 21041 bus latency timer. Figure 3–4 shows the CFLT bit field, and Table 3–8 describes the bit field.

**Figure 3–4 CFLT Configuration Latency Timer Register**

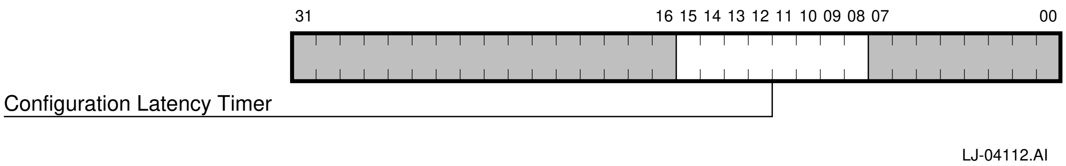



**Table 3–8 CFLT Configuration Latency Timer Register Description**

| Field | Description |
|---|---|
| **15:8** | **Configuration Latency Timer**<br>Specifies, in units of PCI bus clocks, the value of the latency timer of the 21041.<br>When the 21041 asserts frame_l, it enables its latency timer to count.<br>If the 21041 deasserts frame_l prior to count expiration, the content of the latency timer is ignored. Otherwise, after the count expires, the 21041 initiates transaction termination as soon as its gnt_l is deasserted. |

Table 3–9 lists the access rules for the CFLT register.

**Table 3–9 CFLT Access Rules**

| Category | Description |
|---|---|
| Value after hardware reset | 0H |
| Read access rules | – |
| Write access rules | Written once during configuration. |

### 3.2.5 Configuration Base I/O Address Register (CBIO)

The CBIO register specifies the base I/O address for accessing the 21041 CSRs (CSR0 through CSR15). For example, if the CBIO register is programmed to 1000H, the I/O address of CSR15 is equal to CBIO + CSR15-offset for a value of 1078H (Table 3–20).

This register must be initialized prior to accessing any CSR.

Figure 3–5 shows the CBIO bit fields, and Table 3–10 describes the bit fields.

**Figure 3–5 CBIO Configuration Base I/O Address Register**

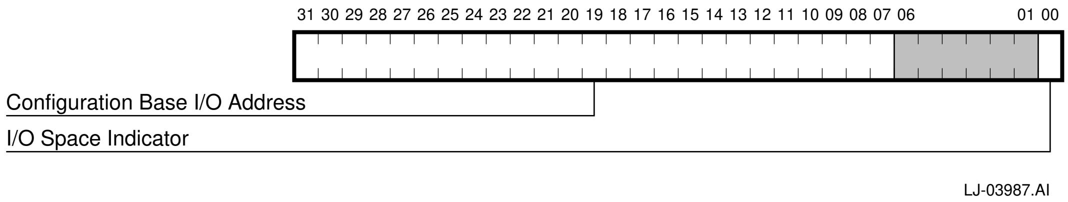



**Table 3–10 CBIO Configuration Base I/O Address Register Description**

| Field | Description |
|---|---|
| **31:7** | **Configuration Base I/O Address**<br>Defines the address assignment mapping of 21041 CSRs. |
| **6:1** | **This field value is 0 when read.** |
| **0** | **I/O Space Indicator**<br>Determines that the register maps into the I/O space. The value in this field is 1. This is a read-only field. |

Table 3–11 lists the access rules for the CBIO register.

**Table 3–11 CBIO Access Rules**

| Category | Description |
|---|---|
| Value after reset | Software reset has no effect. |
| Read access rules | – |
| Write access rules | Written once during configuration. |

### 3.2.6 Configuration Base Memory Address Register (CBMA)

The CBMA register specifies the base memory address for memory accesses to the 21041 CSRs (CSR0 through CSR15).

This register must be initialized prior to accessing any CSR with memory access.

Figure 3–6 shows the CBMA bit fields, and Table 3–12 describes the bit fields.

**Figure 3–6 CBMA Configuration Base Memory Address Register**

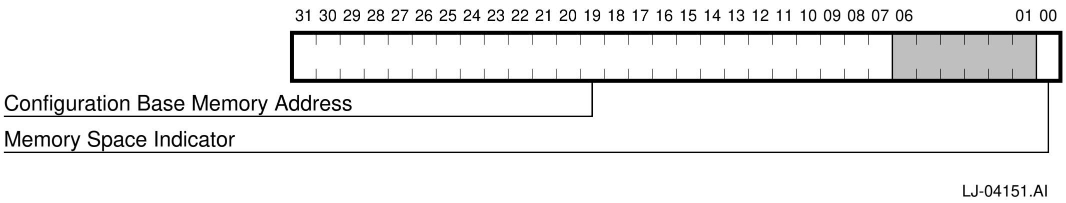



**Table 3–12 CBMA Configuration Base Memory Address Register Description**

| Field | Description |
|---|---|
| **31:7** | **Configuration Base Memory Address**<br>Defines the address assignment mapping of the 21041 CSRs. |
| **6:1** | **This field value is 0 when read.** |
| **0** | **Memory Space Indicator**<br>Determines that the register maps into the memory space. The value in this field is 0. This is a read-only field. |

Table 3–13 lists the access rules for the CBMA register.

**Table 3–13 CBMA Access Rules**

| Category | Description |
|---|---|
| Value after reset | Software reset has no effect. |
| Read access rules | – |
| Write access rules | Written once during configuration. |

## 3.2.7 Expansion ROM Base Address Register (CBER)

The CBER register specifies the base address and gives information about the expansion ROM size. This register must be initialized prior to accessing the expansion ROM.

Figure 3–7 shows the CBER bit fields, and Table 3–14 describes the bit fields.

**Figure 3–7 CBER Expansion ROM Base Address Register**

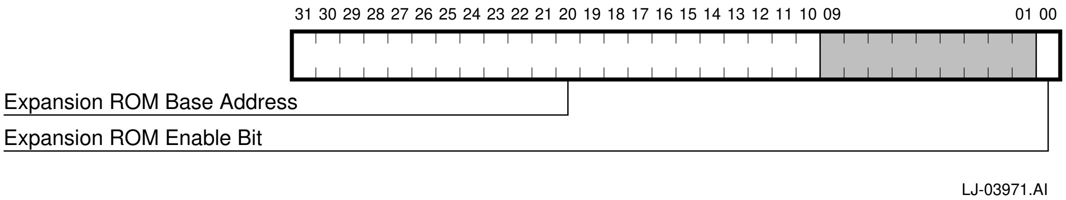



**Table 3–14 CBER Expansion ROM Base Address Register Description**

| Field | Description |
|---|---|
| **31:10** | **Expansion ROM Base Address**<br>Defines the address assignment mapping of the expansion ROM. It also gives information about the expansion ROM size. CBER<17:10> are hardwired to 0 indicating the expansion ROM size to be 256K bytes. |
| **9:1** | **This field value is 0 when read.** |
| **0** | **Expansion ROM Enable Bit**<br>The 21041 responds at accesses to its expansion ROM only if the Memory Space bit (CFCS<1>) and the Expansion ROM Enable bit are set to 1. In addition, the Memory Base Address Register (CBMA) must be written before the ROM Base Address Register (CBER) is written. |

Table 3–15 lists the access rules for the CBER register.

**Table 3–15 CBER Access Rules**

| Category | Description |
|---|---|
| Value after reset | Software reset has no effect. |
| Read access rules | – |
| Write access rules | Written during configuration. |

### 3.2.8 Configuration Interrupt Register (CFIT)

The CFIT register is divided into two sections: the interrupt line and the interrupt pin. CFIT configures both the system's interrupt line and the 21041 interrupt pin connection.

Figure 3–8 shows the CFIT bit fields, and Table 3–16 describes the bit fields.

**Figure 3–8 CFIT Configuration Interrupt Register**

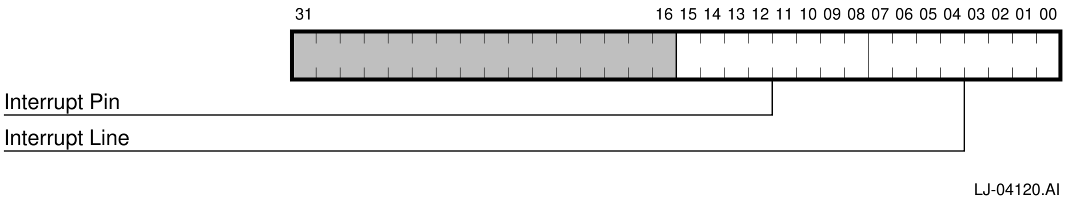



**Table 3–16 CFIT Configuration Interrupt Register Description**

| Field | Description |
|---|---|
| **15:8** | **Interrupt Pin**<br>Indicates which interrupt pin the 21041 uses. The 21041 uses INTA#, and the read value is 01H. |
| **7:0** | **Interrupt Line**<br>Provides interrupt line routing information. The BIOS writes the routing information into this field when it initializes and configures the system.<br>The value in this field indicates which input of the system interrupt controller the 21041's interrupt pin is connected to. The driver can use this information to determine priority and vector information. Values in this field are architecture-specific. |

Table 3–17 lists the access rules for the CFIT register.

**Table 3–17 CFIT Access Rules**

| Category | Description |
|---|---|
| Value after reset | Software reset has no effect. |
| Read access rules | – |
| Write access rules | – |

### 3.2.9 Configuration Driver Area Register (CFDA)

The CFDA register can be used to store driver-specific information during initialization. It has no effect on the 21041 operation.

Figure 3–9 shows the CFDA bit field, and Table 3–18 describes the bit field.

**Figure 3–9 CFDA Configuration Driver Area Register**

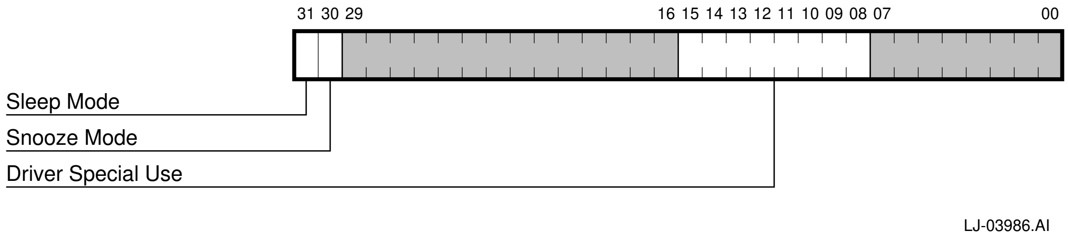



**Table 3–18 CFDA Configuration Driver Area Register Description**

| Field | Description |
|---|---|
| **31** | **Sleep Mode**<br>When set, the 21041 enters the sleep power saving mode. Refer to Section 5.9 for a description of the power saving modes. |
| **30** | **Snooze Mode**<br>When set, the 21041 enters the snooze power saving mode. Refer to Section 5.9 for a description of the power saving modes. |
| **15:8** | **Driver Special Use**<br>Read and write fields for the driver's special use. |

Table 3–19 lists the access rules for the CFDA register.

**Table 3–19 CFDA Access Rules**

| Category | Description |
|---|---|
| Value after reset | Software reset has no effect, except for bit 31 which is 0. |
| Read access rules | – |
| Write access rules | – |

## 3.3 Command and Status Registers

The 21041 contains 16 command and status registers (CSRs), which can be accessed by the host. Table 3–20 lists the CSR registers.

**Table 3–20 CSR Mapping**

| Register | Description | Offset from CSR Base Address (CBIO, CBMA) |
|---|---|---|
| CSR0 | Bus mode register | 00H |
| CSR1 | Transmit poll demand register | 08H |
| CSR2 | Receive poll demand register | 10H |
| CSR3 | Receive list base address register | 18H |
| CSR4 | Transmit list base address register | 20H |
| CSR5 | Status register | 28H |
| CSR6 | Operation mode register | 30H |
| CSR7 | Interrupt mask register | 38H |
| CSR8 | Missed frame counter register | 40H |
| CSR9 | Boot and serial ROMs register | 48H |
| CSR10 | Boot ROM address register | 50H |
| CSR11 | General purpose timer register | 58H |
| CSR12 | SIA status register | 60H |
| CSR13 | SIA connectivity register | 68H |
| CSR14 | SIA transmit receive register | 70H |
| CSR15 | SIA general register | 78H |

The 21041 CSRs are located in the host I/O or memory address space. The CSRs are *quadword*-aligned and can only be accessed using *longword* instructions.

**Note**

- Register access is only longword access; byte accesses to CSR0 through CSR15 are not supported. Accessing a non-longword address register causes UNPREDICTABLE data results.
- Reserved bits must be written with 0. Reserved bits are UNPREDICTABLE on read accesses.
- Retries on second data transactions occur in response to burst I/O accesses.

CSRs are physically located in the chip. The host uses a single instruction to access to a CSR. Most commonly used 21041 features are contained in the CSRs.

### 3.3.1 Host CSRs

There are 12 CSRs (CSR0 through CSR11) used to communicate with the host.

### 3.3.2 Bus Mode Register (CSR0)

CSR0 establishes the bus operating modes. Figure 3–10 shows the CSR0 bit fields, and Table 3–21 describes the bit fields.

**Figure 3–10 CSR0 Bus Mode Register**

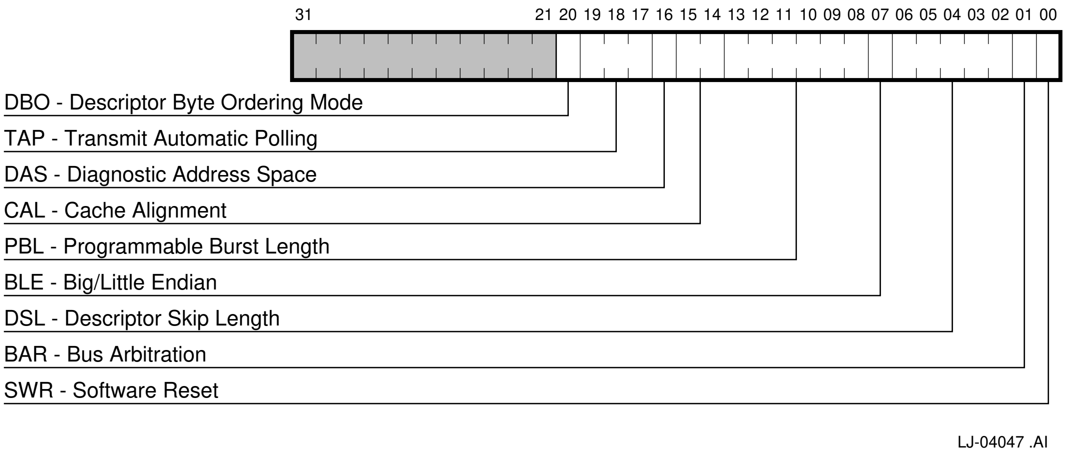



**Table 3–21 CSR0 Bus Mode Register Description**

| Field | Description |
|---|---|
| **20** | **DBO—Descriptor Byte Ordering Mode**<br>When set, the 21041 operates in big endian byte ordering mode for descriptors only.<br>When reset, the 21041 operates in little endian mode. |
| **19:17** | **TAP—Transmit Automatic Polling (read, write)**<br>The 21041 performs a transmit automatic poll demand when programmed to do so, and it is in a suspended state because a transmit buffer is unavailable. Refer to Table 3–22. |
| **16** | **DAS—Diagnostic Address Space (read, write)**<br>When reset, CSR0 through CSR15 are mapped on I/O space and memory space (21041 address space becomes 128 bytes).<br>When set, all 16 CSRs and all diagnostic registers are mapped on I/O and memory space. |

(continued on next page)

**Table 3–21 (Cont.) CSR0 Bus Mode Register Description**

| Field | Description |
|---|---|
| **15:14** | **CAL—Cache Alignment (read, write)**<br>Programmable address boundaries for data burst stop (Table 3–24). If the buffer is not aligned, the 21041 executes the first transfer up to the address boundary, then all transfers are aligned to the specified boundary.<br>**Note:** During initialization, this field must be programmed to the required cache alignment size. Leaving the value of this field at 00 will have unpredictable results. |
| **13:8** | **PBL—Programmable Burst Length (read, write)**<br>Indicates the maximum number of longwords to be transferred in one DMA transaction. If PBL = 0, the 21041 burst is limited only by the amount of data stored in the receive FIFO (at least 16 longwords) or by the amount of free space in the transmit FIFO (at least 16 longwords) before issuing a bus request.<br>The PBL can be programmed with permissible values 0, 1, 2, 4, 8, 16, or 32. After reset, the PBL default value is 0. |
| **7** | **BLE—Big/Little Endian (read, write)**<br>When set, the 21041 operates in big endian byte ordering mode. When reset, the 21041 operates in little endian byte ordering mode.<br>Big endian is applicable only for data buffers.<br>For example, the byte order in little endian of a data buffer is 12345678H, with each digit representing a nibble. In big endian, the byte orientation is 78563412H. |
| **6:2** | **DSL—Descriptor Skip Length (read, write)**<br>Specifies the number of longwords to skip between two descriptors.<br>To improve performance, descriptors can be placed in a separate cache line and do not have to be contiguous. |
| **1** | **BAR—Bus Arbitration (read, write)**<br>Selects the internal bus arbitration between the receive and transmit processes.<br>When set, a round robin arbitration scheme is applied resulting in equal sharing between processes. When reset to 0, the receive process has priority over the transmit process, unless the 21041 is currently transmitting (Section 5.2). |

(continued on next page)

**Table 3–21 (Cont.) CSR0 Bus Mode Register Description**

| Field | Description |
|---|---|
| **0** | **SWR—Software Reset (read, write)**<br>When set, the 21041 resets all internal hardware.<br>When reset, duration should be at least 10 PCI clock cycles. After reset deassertion, the first bus transaction to the 21041 should not be initiated before at least 50 more PCI cycles elapse.<br>Software reset does not affect the configuration area. |

Table 3–22 defines the transmit automatic polling bits.

**Table 3–22 Transmit Automatic Polling Bits**

| CSR0<19:17> | Time Intervals |
|---|---|
| 000 | No transmit automatic polling; CSR1 access should be used to poll the transmit descriptor list. |
| 001 | Transmit automatic polling every 200 microseconds. |
| 010 | Transmit automatic polling every 800 microseconds. |
| 011 | Transmit automatic polling every 1.6 milliseconds. |
| 100 | Transmit automatic polling every 12.8 microseconds. |
| 101 | Transmit automatic polling every 25.6 microseconds. |
| 110 | Transmit automatic polling every 51.2 microseconds. |
| 111 | Transmit automatic polling every 102.4 microseconds. |

Table 3–23 lists the access rules for the CSR0 register.

**Table 3–23 CSR0 Access Rules**

| Category | Description |
|---|---|
| Value after reset | FFF00000H |
| Read access rules | – |
| Write access rules | To write, the transmit and receive processes must be stopped. If one or both of the processes is not stopped, the result is UNPREDICTABLE. |

Table 3–24 defines the cache address alignment bits.

**Table 3–24 Cache Address Alignment Bits**

| CSR0<15:14> | Address Alignment |
|---|---|
| 00 | Not allowed |
| 01 | 8-longword boundary alignment |
| 10 | 16-longword boundary alignment |
| 11 | 32-longword boundary alignment |

### 3.3.3 Transmit Poll Demand Register (CSR1)

Figure 3–11 shows the CSR1 bit fields, and Table 3–25 describes the bit fields.

**Figure 3–11 CSR1 Transmit Poll Demand Register**

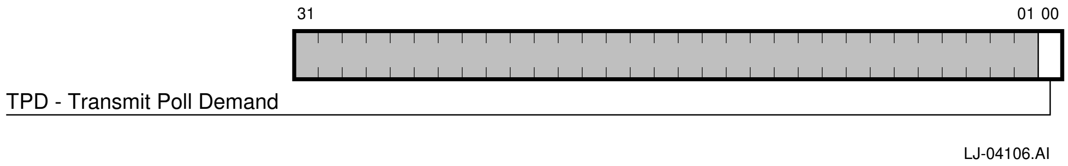



**Table 3–25 CSR1 Transmit Poll Demand Register Description**

| Field | Description |
|---|---|
| **0** | **TPD—Transmit Poll Demand (write)**<br>When written with any value, the 21041 checks for frames to be transmitted. If no descriptor is available, the transmit process moves to the suspended state and CSR5<2> is not asserted. If the descriptor is available, the transmit process resumes.<br>If a poll demand is issued when the transmit process is in the running state after the 21041 has prefetched the next descriptor and found it to be host-owned, the 21041 rechecks the descriptor list in host memory before moving to the suspended state. |

Table 3–26 lists the access rules for the CSR1 register.

**Table 3–26 CSR1 Access Rules**

| Category | Description |
|---|---|
| Value after reset | FFFFFFFFH |
| Read access rules | – |
| Write access rules | All bits except for bit 0 should be written with 0. |

### 3.3.4 Receive Poll Demand Register (CSR2)

Figure 3–12 shows the CSR2 bit field, and Table 3–27 describes the bit field.

**Figure 3–12 CSR2 Receive Poll Demand Register**

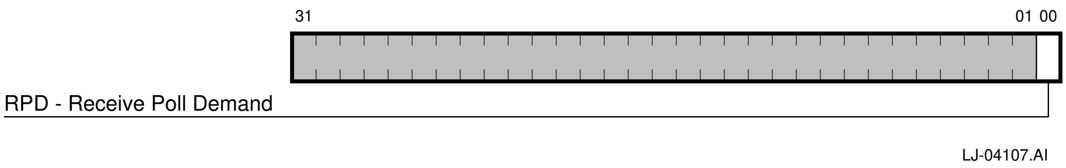



**Table 3–27 CSR2 Receive Poll Demand Register Description**

| | |
|---|---|
| **0** | **RPD—Receive Poll Demand (write)**<br>When written with any value, the 21041 checks for receive descriptors to be acquired. If no descriptor is available, the receive process moves to the suspended state and CSR5<7> is not asserted. If the descriptor is available, the receive process resumes.<br>If a poll demand is issued when the receive process is in the running state after the 21041 has prefetched the next descriptor and found it to be host-owned, the 21041 rechecks the descriptor list in host memory before moving to the suspended state. |

Table 3–28 lists the access rules for the CSR2 register.

**Table 3–28 CSR2 Access Rules**

| Category | Description |
|---|---|
| Value after reset | FFFFFFFFH |
| Read access rules | – |
| Write access rules | All bits except for bit 0 should be written with 0. |

### 3.3.5 Descriptor List Address Registers (CSR3, CSR4)

The CSR3 descriptor list address register is used for receive buffer descriptors, and the CSR4 descriptor list address register is used for transmit buffer descriptors. Both registers are used to point the 21041 to the start of the appropriate descriptor list.

Figure 3–13 shows the CSR3 bit field, and Table 3–29 describes the bit field.

**Note**

The descriptor lists reside in *physical* memory space and must be *longword*-aligned. The 21041 behaves unpredictably when the lists are not longword-aligned.

Writing to either CSR3 or CSR4 is permitted only when its respective process is in the stopped state. When stopped, the CSR3 and CSR4 registers must be written before the respective START command is given (Section 3.3.7).

**Figure 3–13 CSR3 Receive List Base Address Register**

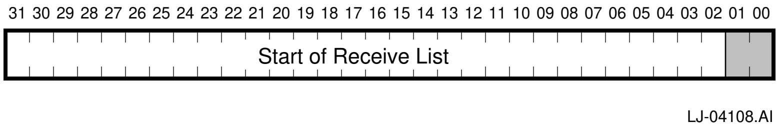



**Table 3–29 CSR3 Receive List Base Address Register Description**

| Field | Description |
|---|---|
| **31:2** | **Start of Receive List (read, write)** |
| **1:0** | **Must be 00. (read, write)** |

Figure 3–14 shows the CSR4 bit field, and Table 3–30 describes the bit field.

**Figure 3–14 CSR4 Transmit List Base Address Register**

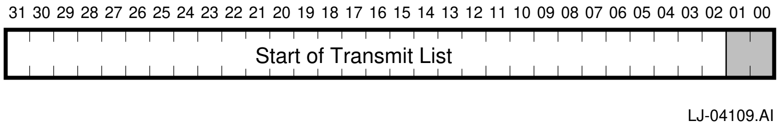



**Table 3–30 CSR4 Transmit List Base Address Register Description**

| Field | Description |
|---|---|
| **31:2** | **Start of Transmit List (read, write)** |
| **1:0** | **Must be 00. (read, write)** |

Table 3–31 lists the access rules for the CSR3 register, and Table 3–32 lists the access rules for the CSR4 register.

**Table 3–31 CSR3 Access Rules**

| Category | Description |
|---|---|
| Value after reset | UNPREDICTABLE |
| Read access rules | – |
| Write access rules | Receive process stopped |

**Table 3–32 CSR4 Access Rules**

| Category | Description |
|---|---|
| Value after reset | UNPREDICTABLE |
| Read access rules | – |
| Write access rules | Transmit process stopped |

### 3.3.6 Status Register (CSR5)

The CSR5 register contains all the status bits that the 21041 reports to the host. CSR5 is usually read by the driver during interrupt service routine or polling. Most of the fields in this register cause the host to be interrupted. CSR5 bits are not cleared when read. Writing 1 to these bits clears them; writing 0 has no effect. Each field can be masked (Section 3.3.8).

Figure 3–15 shows the CSR5 bit fields, and Table 3–33 describes the bit fields.

**Figure 3–15 CSR5 Status Register**

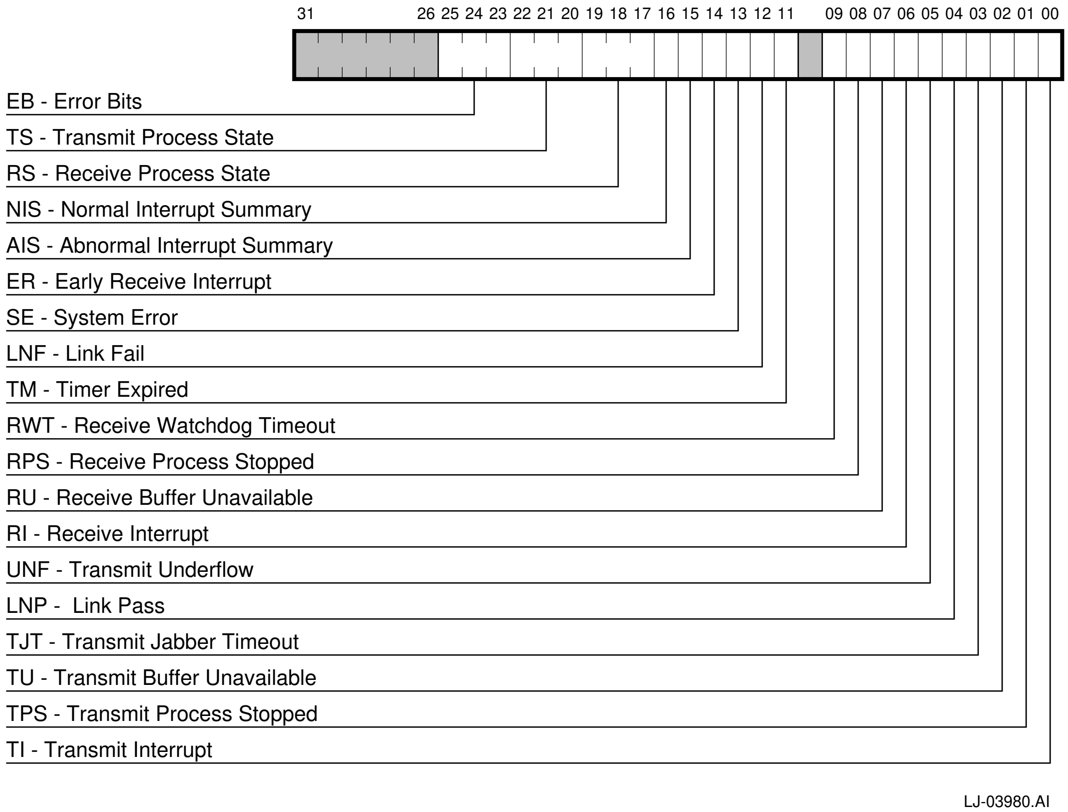


**Table 3–33 CSR5 Status Register Description**

| Field | Description |
|---|---|
| **25:23** | **EB—Error Bits (read)**<br>Indicates the type of error that caused a system error. Valid only when system error CSR5<13> is set (Table 3–34). This field does not generate an interrupt. |
| **22:20** | **TS—Transmit Process State (read)**<br>Indicates the state of the transmit process (Table 3–35). This field does not generate an interrupt. |
| **19:17** | **RS—Receive Process State (read)**<br>Indicates the state of the receive process (Table 3–36). This field does not generate an interrupt. |
| **16** | **NIS—Normal Interrupt Summary (read, write)**<br>Normal interrupt summary bit. Its value is the logical OR of<br>CSR5<0>—Transmit interrupt<br>CSR5<2>—Transmit buffer unavailable<br>CSR5<6>—Receive interrupt<br>CSR5<11>—Timer expired<br>CSR5<14>—Early receive interrupt<br>Unmasked bits affect only the normal interrupt summary CSR5<16> bit. |
| **15** | **AIS—Abnormal Interrupt Summary (read, write)**<br>Abnormal interrupt summary bits. Its value is the logical OR of<br>CSR5<1>—Transmit process stopped<br>CSR5<3>—Transmit jabber timeout<br>CSR5<4>—Link pass<br>CSR5<5>—Transmit underflow<br>CSR5<7>—Receive buffer unavailable<br>CSR5<8>—Receive process stopped<br>CSR5<9>—Receive watchdog timeout<br>CSR5<12>—Link fail<br>CSR5<13>—System error<br>Unmasked bits affect only the abnormal interrupt summary CSR5<15> bit. |
| **14** | **ER—Early Receive Interrupt (read, write)**<br>Indicates the completion of processing the first receive buffer. CSR5<14> is asserted when writing to the first receive buffer is complete. CSR5<14> and CSR5<6> will never be asserted together. CSR5<6> automatically clears CSR5<14>. |

(continued on next page)

**Table 3–33 (Cont.) CSR5 Status Register Description**

| Field | Description |
|---|---|
| **13** | **SE—System Error (read, write)**<br>Indicates that a system error occurred (Table 3–34). When this bit is set, all DMA accesses stop until the bit is cleared by software reset or by writing 1. |
| **12** | **LNF—Link Fail (read, write)**<br>Indicates a transition to the link fail state in the twisted-pair port. See link fail status CSR12<2>.<br>This bit is meaningful only when CSR14<8>, Receive Squelch Enable, is set and CSR13<3>, 10BASE-T or AUI, is 0 (10BASE-T mode). This bit is also set as a result of setting CSR15<10>, Force Link Fail. |
| **11** | **TM—Timer Expired (read, write)**<br>Indicates the expiration of the general-purpose timer. See Section 3.3.12. |
| **9** | **RWT—Receive Watchdog Timeout (read, write)**<br>Indicates that the receive watchdog timer expired, and another node is babbling on the network. Current frame reception aborts while length error RDES0<14> and last descriptor RDES0<8> assert. Receive interrupt CSR5<6> also asserts, and the receive process remains in the running state. |
| **8** | **RPS—Receive Process Stopped (read, write)**<br>Indicates that the receive process is stopped. Table 5–2 explains the receive process state transitions. |
| **7** | **RU—Receive Buffer Unavailable (read, write)**<br>Indicates that the next descriptor in the receive list is owned by the host and cannot be acquired by the 21041. The reception process is suspended. To resume processing receive descriptors, the host should change the ownership of the descriptor and might issue a receive poll demand command. If no receive poll demand is issued, the reception process resumes when the next recognized incoming frame is received.<br>After the first assertion, CSR5<7> does not assert for any subsequent not owned receive descriptors fetches. CSR5<7> asserts only when the previous receive descriptor was owned by the 21041. |

(continued on next page)

**Table 3–33 (Cont.) CSR5 Status Register Description**

| Field | Description |
|---|---|
| **6** | **RI—Receive Interrupt (read, write)**<br>Indicates the completion of a frame reception. Specific frame status information has been posted in the descriptor. The reception process remains in the running state. |
| **5** | **UNF—Transmit Underflow (read, write)**<br>Indicates that the transmit FIFO had an underflow condition during the packet transmission. The transmit process is placed in the suspended state, and underflow error TDES0<1> is set. |
| **4** | **LNP—Link Pass (read, write)**<br>Indicates that the 10BASE-T Link Integrity Test has completed successfully, after the link was down. This bit is also set as a result of writing 0 to CSR14<12>, Link Test Enable. |
| **3** | **TJT—Transmit Jabber Timeout (read, write)**<br>Indicates that the transmit jabber timer expired, meaning that the 21041 transmitter was babbling. The transmission process is aborted and placed in the stopped state. This event causes the transmit jabber timeout TDES0<14> flag to assert. |
| **2** | **TU—Transmit Buffer Unavailable (read, write)**<br>Indicates that the next descriptor on the transmit list is owned by the host and cannot be acquired by the 21041. The transmission process is suspended. Table 5–3 explains the transmit process state transitions. To resume processing transmit descriptors, the host should change the ownership bit of the descriptor, then issue a transmit poll demand command, unless transmit automatic polling (Table 3–22) is enabled. |
| **1** | **TPS—Transmit Process Stopped (read, write)**<br>Asserts when the transmit process enters the stopped state. |
| **0** | **TI—Transmit Interrupt (read, write)**<br>Indicates that a frame transmission was completed, while TDES1<31> is asserted in the first descriptor of the frame. |

Table 3–34 lists the bit codes for the bus error bits.

**Table 3–34 Bus Error Bits**

| CSR5<25:23> | Process State |
|---|---|
| 000 | Parity error |
| 001 | Master abort |
| 010 | Target abort |
| 011 | Reserved |
| 1XX | Reserved |

Table 3–35 lists the bit codes for the transmit process state.

**Table 3–35 Transmit Process State**

| CSR5<22:20> | Process State |
|---|---|
| 000 | Stopped—RESET command or transmit jabber expired. |
| 001 | Running—Fetch transmit descriptor. |
| 010 | Running—Wait for end of transmission. |
| 011 | Running—Read buffer from memory, and queue the data into the transmit FIFO. |
| 100 | Reserved. |
| 101 | Running—Setup packet. |
| 110 | Suspended—Transmit FIFO underflow or an unavailable transmit descriptor. |
| 111 | Running—Close transmit descriptor. |

Table 3–36 lists the bit codes for the receive process state.

**Table 3–36 Receive Process State**

| CSR5<19:17> | Process State |
|---|---|
| 000 | Stopped—RESET or STOP RECEIVE command. |
| 001 | Running—Fetch receive descriptor. |
| 010 | Running—Check for end-of-receive packet before prefetch of next descriptor. |
| 011 | Running—Wait for receive packet. |
| 100 | Suspended—Unavailable receive buffer. |
| 101 | Running—Close receive descriptor. |
| 110 | Running—Flush the current frame from the receive FIFO because of unavailable receive buffer. |
| 111 | Running—Queue the receive frame from the receive FIFO into the receive buffer. |

Table 3–37 lists the access rules for the CSR5 register.

**Table 3–37 CSR5 Access Rules**

| Category | Description |
|---|---|
| Value after reset | FC000000H |
| Read access rules | – |
| Write access rules | CSR5 bits 0 through 16 are cleared by writing 1. Writing 0 to these bits has no effect. Writing to CSR5 bits 17 through 25 has no effect. |

### 3.3.7 Operation Mode Register (CSR6)

The CSR6 register establishes the receive and transmit operating modes and commands. CSR6 should be the last CSR to be written as part of initialization. Figure 3–16 shows the CSR6 bit fields, and Table 3–38 describes the bit fields.

**Figure 3–16 CSR6 Operating Mode Register**

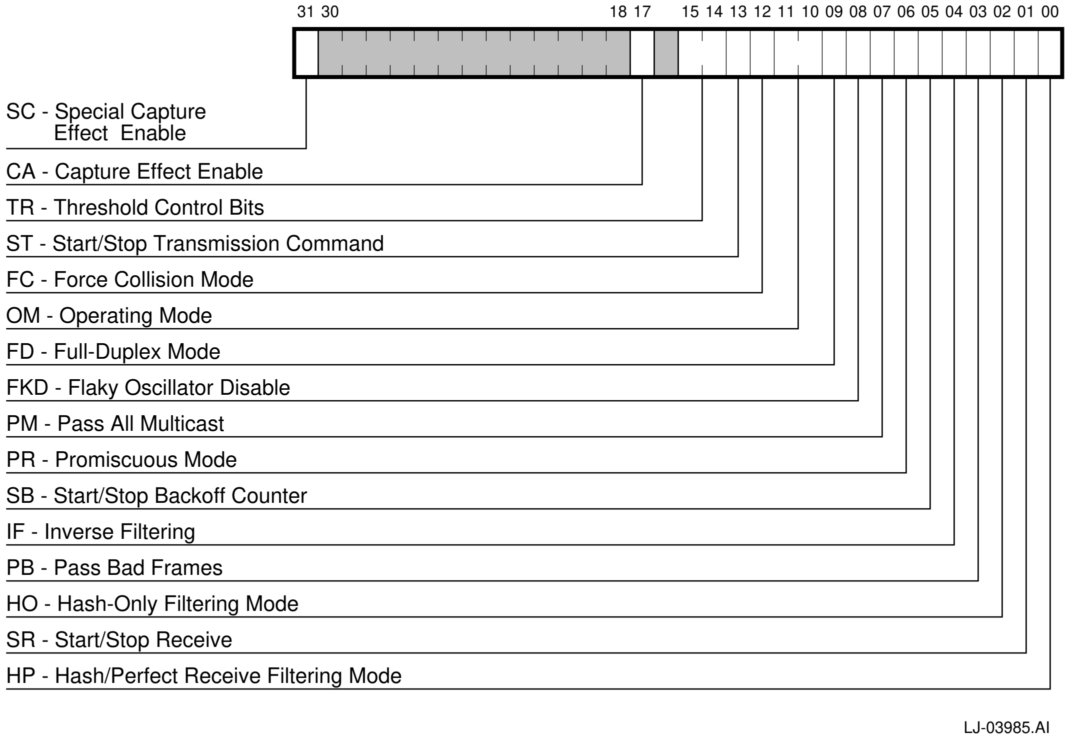


**Table 3–38 CSR6 Operating Mode Register Description**

| Field | Description |
|---|---|
| **31** | **SC—Special Capture Effect Enable (read, write)**<br>When set, enables the enhanced resolution of capture effect on the network. See Section 7.4.7. It is recommended that the SC bit is set together with CSR6<17>.<br>When clear, the 21041 disables the enhanced resolution of capture effect on the network. |
| **17** | **CA—Capture Effect Enable (read, write)**<br>When set, enables the resolution of the capture effect on the network. See Section 7.4.7.<br>When reset, the 21041 disables the resolution of capture effect on the network. |
| **15:14** | **TR—Threshold Control Bits (read, write)**<br>Controls the selected threshold level for the 21041 transmit FIFO. Four threshold levels are allowed (Table 3–39).<br>The threshold value has a direct impact on the 21041 bus arbitration scheme (Section 5.2).<br>Transmission starts when the frame size within the transmit FIFO is larger than the threshold. Full frames with a length less than the threshold are also transmitted.<br>The transmit process must be in the stopped state to change these bits CSR6<15:14>. |
| **13** | **ST—Start/Stop Transmission Command (read, write)**<br>When set, the transmission process is placed in the running state, and the 21041 checks the transmit list at the current position for a frame to be transmitted.<br>Descriptor acquisition is attempted either from the current position in the list, which is the transmit list base address set by CSR4, or from the position retained when the transmit process was previously stopped. If no descriptor can be acquired, the transmit process enters the suspended state.<br>If the current descriptor is not owned by the 21041, the transmission process enters the suspended state, and transmit buffer unavailable CSR5<2> is set. The start transmission command is honored only when the transmission process is stopped. If the command is issued before setting CSR4, the 21041 will behave unpredictably.<br>When reset, the transmission process is placed in the stopped state after completing the transmission of the current frame. The next descriptor position in the transmit list is saved and becomes the current position when transmission is restarted.<br>The stop transmission command is honored only when the transmission process is in either the running or suspended state (Table 5–3). |

(continued on next page)

**Table 3–38 (Cont.) CSR6 Operating Mode Register Description**

| Field | Description |
|---|---|
| **12** | **FC—Force Collision Mode (read, write)**<br>Allows the collision logic to be tested. Meaningful only in internal loopback mode. When set, a collision is forced during the next transmission attempt. This results in 16 transmission attempts with excessive collision reported in the transmit descriptor TDES0<8>. |
| **11:10** | **OM—Operating Mode (read, write)**<br>Selects the 21041 main mode of operation in the MAC level (Table 3–41). |
| **9** | **FD—Full-Duplex Mode (read, write)**<br>This bit is used by the driver to activate the 21041 in full-duplex mode (Section 5.8). When the autonegotiation algorithm is used (CSR14<7> autonegotiation enable is set), the 21041 operates in full-duplex mode only if the negotiation results permit full-duplex operation.<br>When set and autonegotiation is disabled, the 21041 operates in full-duplex mode. The 21041 transmit and receive functions occur simultaneously.<br>Setting the 21041 to operate in full-duplex mode is allowed only if the transmit and receive processes are in the stopped state, and if the start/stop receive (CSR6<1>) and start/stop transmission commands (CSR6<13>) are both set to 0.<br>While in full-duplex mode, heartbeat check is disabled, heartbeat fail TDES0<7> should be ignored, and internal loopback is not allowed. |
| **8** | **FKD—Flaky Oscillator Disable (read, write)**<br>When set, indicates that the internal flaky oscillator is disabled; pseudo-random numbers are chosen instead of fully random numbers. This bit is set only for diagnostic purposes. |
| **7** | **PM—Pass All Multicast (read, write)**<br>When set, indicates that all the incoming frames with a multicast destination address (first bit in the destination address field is 1) are received. Incoming frames with physical address destinations are filtered according to the CSR6<0> bit. |
| **6** | **PR—Promiscuous Mode (read, write)**<br>When set, indicates that any incoming valid frame is received, regardless of its destination address.<br>After reset, the 21041 wakes up in promiscuous mode. |

(continued on next page)

**Table 3–38 (Cont.) CSR6 Operating Mode Register Description**

| Field | Description |
|---|---|
| **5** | **SB—Start/Stop Backoff Counter (read, write)**<br>When set, indicates that the internal backoff counter stops counting when any carrier activity is detected. The 21041 backoff counter resumes when the carrier drops. The earliest the 21041 starts its transmission is 9.6 microseconds after carrier deassertion.<br>When reset, the internal backoff counter is not affected by the carrier activity. |
| **4** | **IF—Inverse Filtering (read)**<br>When set, the 21041 operates in an inverse filtering mode (Table 4–8). This bit can be written by a setup frame descriptor. Inverse filtering should be used only with perfect filtering. |
| **3** | **PB—Pass Bad Frames (read, write)**<br>When set, the 21041 operates in pass bad frame mode. All incoming frames that passed the address filtering are received, including runt frames, collided fragments, or truncated frames caused by FIFO overflow.<br>If any received bad frames are required, promiscuous mode (CSR6<6>) should be set to 1. |
| **2** | **HO—Hash-Only Filtering Mode (read)**<br>When set, the 21041 operates in an imperfect address filtering mode for both physical and multicast addresses (Table 4–8). This bit can be written by a setup frame descriptor. |
| **1** | **SR—Start/Stop Receive (read, write)**<br>When set, the receive process is placed in the running state. The 21041 attempts to acquire a descriptor from the receive list and processes incoming frames.<br>Descriptor acquisition is attempted from the current position in the list, which is the address set by CSR3 or the position retained when the receive process was previously stopped. If no descriptor is owned by the 21041, the receive process enters the suspended state and receive buffer unavailable CSR5<7> sets.<br>The start reception command is honored only when the reception process has stopped. If the command was issued before setting CSR3, the 21041 behaves unpredictably.<br>When cleared, the receive process enters the stopped state after completing the reception of the current frame. The next descriptor position in the receive list is saved, and becomes the current position after the receive process is restarted. The stop reception command is honored only when the receive process is in running or suspended state (Section 5.5.4). |

(continued on next page)

**Table 3–38 (Cont.) CSR6 Operating Mode Register Description**

| Field | Description |
|---|---|
| 0 | **HP—Hash/Perfect Receive Filtering Mode (read)**<br>When reset, the 21041 does a perfect address filter of incoming frames according to the addresses specified in the setup frame (Table 4–8).<br>When set, the 21041 does imperfect address filtering of the incoming frame according to the hash table specified in the setup frame. This bit can be written by a setup frame descriptor. |

Table 3–39 lists the threshold values in bytes.

**Table 3–39 Transmit Threshold**

| CSR6<15:14> | Threshold (Bytes) |
|---|---|
| 00 | 72 |
| 01 | 96 |
| 10 | 128 |
| 11 | 160 |

Table 3–40 lists the codes to determine the filtering mode.

**Table 3–40 Filtering Mode**

| CSR6<7> | CSR6<6> | CSR6<4> | CSR6<2> | CSR6<0> | Filtering Mode |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 16 perfect filtering |
| 0 | 0 | 0 | 0 | 1 | 512-bit hash + 1 perfect filtering |
| 0 | 0 | 0 | 1 | 0 | Not applicable |
| 0 | 0 | 0 | 1 | 1 | 512-bit hash for multicast and physical addresses |
| 0 | 0 | 1 | 0 | 0 | Inverse filtering |
| 0 | 0 | 1 | 0 | 1 | Not applicable |
| 0 | 0 | 1 | 1 | 0 | Not applicable |
| 0 | 0 | 1 | 1 | 1 | Not applicable |
| 0 | 1 | 0 | 0 | 0 | Promiscuous |
| 0 | 1 | 0 | 0 | 1 | Promiscuous |
| 0 | 1 | 0 | 1 | 0 | Not applicable |
| 0 | 1 | 0 | 1 | 1 | Promiscuous |
| 0 | 1 | 1 | 0 | 0 | Not used |
| 0 | 1 | 1 | 0 | 1 | Not used |
| 0 | 1 | 1 | 1 | 0 | Not used |
| 0 | 1 | 1 | 1 | 1 | Not used |
| 1 | 0 | 0 | 0 | 0 | Pass all multicast |
| 1 | 0 | 0 | 0 | 1 | Pass all multicast |
| 1 | 0 | 0 | 1 | 0 | Not applicable |
| 1 | 0 | 0 | 1 | 1 | Pass all multicast |
| 1 | 0 | 1 | 0 | 0 | Not used |
| 1 | 0 | 1 | 0 | 1 | Not applicable |

(continued on next page)

**Table 3–40 (Cont.) Filtering Mode**

| CSR6<7> | CSR6<6> | CSR6<4> | CSR6<2> | CSR6<0> | Filtering Mode |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | Not applicable |
| 1 | 0 | 1 | 1 | 1 | Not applicable |
| 1 | 1 | 0 | 0 | 0 | Promiscuous |
| 1 | 1 | 0 | 0 | 1 | Promiscuous |
| 1 | 1 | 0 | 1 | 0 | Not applicable |
| 1 | 1 | 0 | 1 | 1 | Promiscuous |
| 1 | 1 | 1 | 0 | 0 | Not used |
| 1 | 1 | 1 | 0 | 1 | Not applicable |
| 1 | 1 | 1 | 1 | 0 | Not applicable |
| 1 | 1 | 1 | 1 | 1 | Not applicable |

Table 3–41 lists the codes of the MAC level operation mode.

**Table 3–41 Operation Mode**

| CSR6<11:10> | Mode | Description |
|---|---|---|
| 00 | Normal | During transmissions, carrier and collision indication is sensed by the receive MAC but receive data is ignored. |
| 01 | Internal loopback in the MAC level | During transmissions, carrier indication directly from the transmit MAC is sensed by the receive MAC. Transmitted data is looped-back into the receive MAC. Collision indication is ignored. |
| 10 | Loopback, external to the MAC level | During transmissions, carrier indication and data is looped-back to the receive MAC from the physical level. Collision indication is also sensed by the receive MAC. |
| 11 | Not used | |

Table 3–42 describes the only conditions that permit change to a field when modifying values to CSR6.

**Table 3–42 CSR6 Access Rules**

| Category | Description |
|---|---|
| Value after reset | FFFC0040H |
| Read access rules | – |
| Write access rules | Any write should be at least 5-microseconds after CSR13 write operation. |
| * CSR6<3> | Receive process stopped |
| * CSR6<5> | Receive and transmit processes stopped |
| * CSR6<7:6> | Receive process stopped |
| * CSR6<8> | Transmit process stopped |
| * CSR6<9> | Receive and transmit process stopped and SIA in reset state |
| * CSR6<11:10> | Receive and transmit processes stopped |
| * CSR6<12> | Receive and transmit processes stopped, internal_loopback mode |
| * CSR6<15:14> | Transmit process stopped |
| * CSR6<16> | Receive and transmit processes stopped |
| * CSR6<17> | Receive and transmit processes stopped |
| * Start_Receive CSR6<1>=1 | CSR3 initialized |
| * Start_Transmit CSR6<13>=1 | CSR4 initialized |
| * Stop_Receive CSR6<1>=0 | Receive running or suspended |
| * Stop_Transmit CSR6<13>=0 | Transmit running or suspended |

## 3.3.8 Interrupt Mask Register (CSR7)

The CSR7 register masks the interrupts reported by CSR5 (Section 3.3.6). Setting a bit to 1 enables a corresponding interrupt. After a hardware or software reset, all interrupts are disabled. Figure 3–17 shows the CSR7 bit fields, and Table 3–43 describes the bit fields.

**Figure 3–17 CSR7 Interrupt Mask Register**

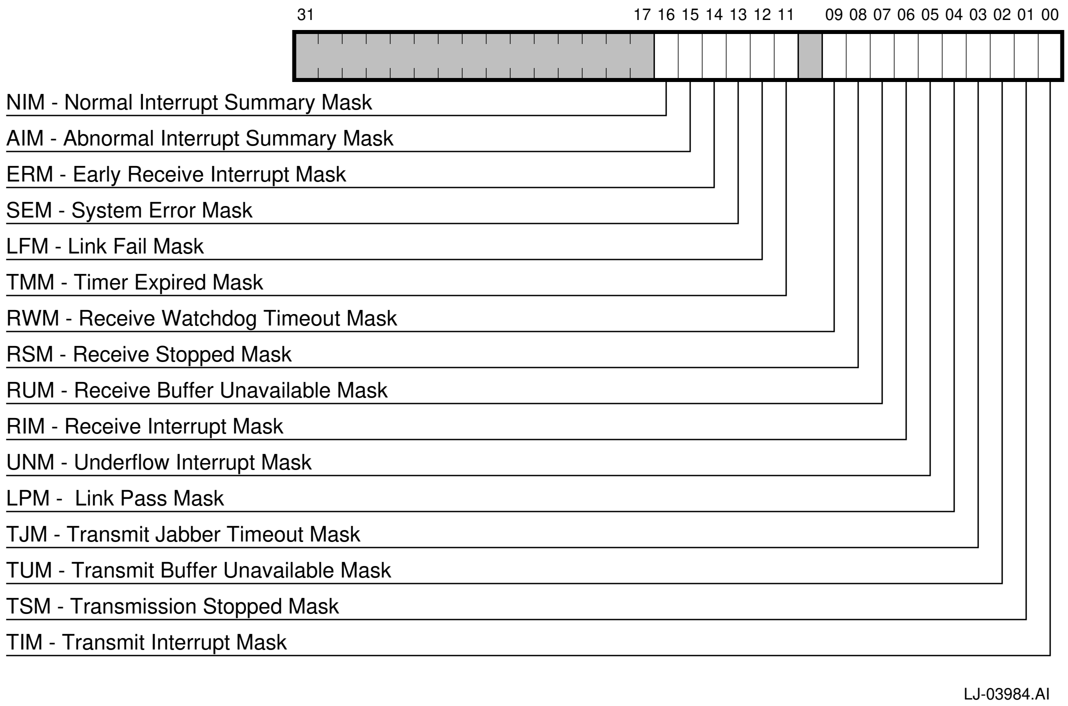


**Table 3–43 CSR7 Interrupt Mask Register Description**

| Field | Description |
|---|---|
| **16** | **NIM—Normal Interrupt Summary Mask (read, write)**<br>When set, normal interrupt is posted.<br>When reset, no normal interrupt is posted. This bit (CSR7<16>) masks the following bits:<br>CSR5<0>—Transmit interrupt<br>CSR5<2>—Transmit buffer unavailable<br>CSR5<6>—Receive interrupt<br>CSR5<11>—Timer expired<br>CSR5<14>—Early receive interrupt |
| **15** | **AIM—Abnormal Interrupt Summary Mask (read, write)**<br>When set, abnormal interrupt is posted.<br>When reset, no abnormal interrupt is posted. This bit CSR7<15> masks the following bits:<br>CSR5<1>—Transmit process stopped<br>CSR5<3>—Transmit jabber timeout<br>CSR5<4>—Link pass<br>CSR5<5>—Transmit underflow<br>CSR5<7>—Receive buffer unavailable<br>CSR5<8>—Receive process stopped<br>CSR5<9>—Receive watchdog timeout<br>CSR5<12>—Link fail<br>CSR5<13>—System error |
| **14** | **ERM—Early Receive Interrupt Mask (read, write)**<br>When set together with normal interrupt summary mask (CSR7<16>) and early receive interrupt bit (CSR5<14>), the interrupt is posted.<br>When reset and the early receive interrupt bit (CSR5<14>) is set, interrupt posting is disabled. |
| **13** | **SEM—System Error Mask (read, write)**<br>When set together with abnormal interrupt summary mask (CSR7<15>) and system error (CSR5<13>), the interrupt is posted. |

(continued on next page)

**Table 3–43 (Cont.) CSR7 Interrupt Mask Register Description**

| Field | Description |
|---|---|
| | When reset and system error (CSR5<13>) is set, the interrupt posting is disabled. |
| **12** | **LFM—Link Fail Mask (read, write)**<br>When set together with abnormal interrupt summary mask (CSR7<15>) and link fail (CSR5<12>), the interrupt is posted.<br>When reset and link fail (CSR5<12>) is set, the interrupt posting is disabled. |
| **11** | **TMM—Timer Expired Mask (read, write)**<br>When set together with normal interrupt summary mask (CSR7<16>) and timer expired bit (CSR5<11>), the interrupt is posted.<br>When reset and the timer expired bit (CSR5<11>) is set, interrupt posting is disabled. |
| **9** | **RWM—Receive Watchdog Timeout Mask (read, write)**<br>When set together with abnormal interrupt summary mask (CSR7<15>) and receive watchdog timeout (CSR5<9>), the interrupt is posted.<br>When reset and receive watchdog timeout (CSR5<9>) is set, the interrupt posting is disabled. |
| **8** | **RSM—Receive Stopped Mask (read, write)**<br>When set together with abnormal interrupt summary mask (CSR7<15>) and receive stopped (CSR5<8>), the interrupt is posted.<br>When reset and receive stopped (CSR5<8>) is set, the interrupt posting is disabled. |
| **7** | **RUM—Receive Buffer Unavailable Mask (read, write)**<br>When set together with abnormal interrupt summary mask (CSR7<15>) and receive buffer unavailable (CSR5<7>), the interrupt is posted.<br>When reset and receive buffer unavailable (CSR5<7>) is set, the interrupt posting is disabled. |
| **6** | **RIM—Receive Interrupt Mask (read, write)**<br>When set together with normal interrupt summary mask (CSR7<16>) and receive interrupt bit (CSR5<6>), the interrupt is posted.<br>When reset and receive interrupt (CSR5<6>) is set, the interrupt posting is disabled. |

(continued on next page)

**Table 3–43 (Cont.) CSR7 Interrupt Mask Register Description**

| Field | Description |
|---|---|
| **5** | **UNM—Underflow Interrupt Mask (read, write)**<br>When set together with the abnormal interrupt summary mask (CSR7<15>) and transmit underflow (CSR5<5>), the interrupt is posted.<br>When reset and transmit underflow (CSR5<5>) are set, interrupt posting is disabled. |
| **4** | **LPM—Link Pass Mask (read, write)**<br>When set together with the abnormal interrupt summary mask (CSR7<15>) and link pass (CSR5<4>), the interrupt is posted.<br>When reset and link pass (CSR5<4>) is set, interrupt posting is disabled. |
| **3** | **TJM—Transmit Jabber Timeout Mask (read, write)**<br>When set together with abnormal interrupt summary mask (CSR7<15>) and transmit jabber timeout (CSR5<3>), the interrupt is posted.<br>When reset and transmit jabber timeout (CSR5<3>) is set, the interrupt posting is disabled. |
| **2** | **TUM—Transmit Buffer Unavailable Mask (read, write)**<br>When set together with normal interrupt summary mask (CSR7<16>) and transmit buffer unavailable (CSR5<2>), the interrupt is posted.<br>When reset and transmit buffer unavailable (CSR5<2>) is set, the interrupt posting is disabled. |
| **1** | **TSM—Transmission Stopped Mask (read, write)**<br>When set together with abnormal interrupt summary mask (CSR7<15>) and transmission stopped (CSR5<1>), the interrupt is posted.<br>When reset and transmission stopped (CSR5<1>) is set, the interrupt posting is disabled. |
| **0** | **TIM—Transmit Interrupt Mask (read, write)**<br>When set together with normal interrupt summary mask (CSR7<16>) and transmit interrupt (CSR5<0>), the interrupt is posted.<br>When reset and transmit interrupt (CSR5<0>) is set, the interrupt posting is disabled. |

Table 3–44 lists the access rules for the CSR7 register.

**Table 3–44 CSR7 Access Rules**

| Category | Description |
|---|---|
| Value after reset | FFFE0000H |
| Read access rules | – |
| Write access rules | – |

### 3.3.9 Missed Frame Counter Register (CSR8)

Figure 3–18 shows the CSR8 bit fields, and Table 3–45 describes the bit fields.

**Figure 3–18 CSR8 Missed Frame Counter Register**

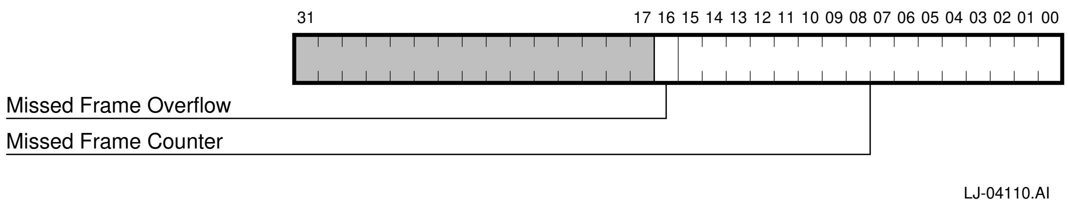



**Table 3–45 CSR8 Missed Frame Counter Register Description**

| Field | Description |
|---|---|
| **16** | **Missed Frame Overflow (read)**<br>Sets when the missed frame counter overflows; resets when CSR8 is read. |
| **15:0** | **Missed Frame Counter (read)**<br>Indicates the number of frames discarded because no host receive descriptors were available. The counter clears when read. |

Table 3–46 lists the access rules for the CSR8 register.

**Table 3–46 CSR8 Access Rules**

| Category | Description |
|---|---|
| Value after reset | FFFE0000H |
| Read access rules | – |
| Write access rules | – |

## 3.3.10 Boot and Ethernet ROMs Register (CSR9)

The CSR9 register provides an interface to the boot ROM and the serial ROM. It is used to select the device, and it contains the commands and data to be read from and stored in the boot ROM and the serial ROM.

Figure 3–19 shows the boot and Ethernet ROMs register, and Table 3–47 describes the register bit fields.

**Figure 3–19 CSR9 Boot and Ethernet ROMs Register**

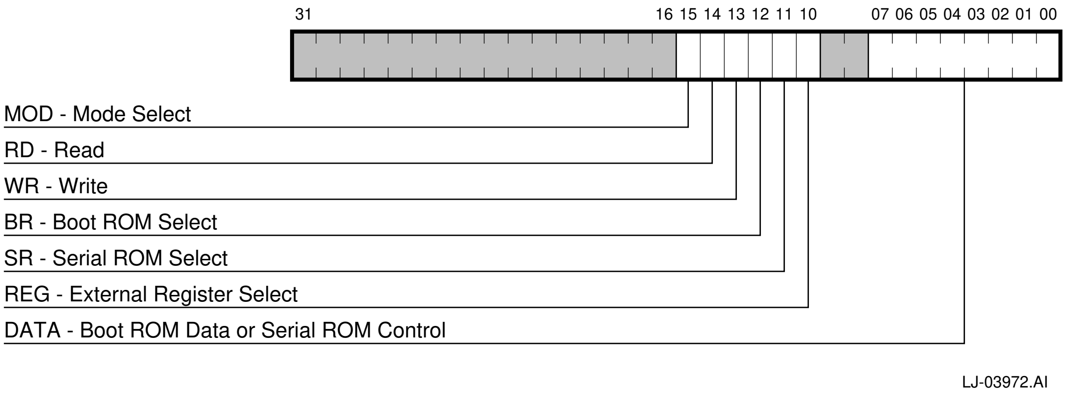


**Table 3–47 CSR9 Boot and Ethernet ROMs Register Description**

| Field | Description | |
|---|---|---|
| **15** | **MOD—Mode Select** | |
| | Specifies the mode selection. This bit is connected to pin Mode_Select (inversed). | |
| | When set, the 21041 operates in mode 0. When clear, the 21041 operates in mode 1 (read only). | |
| **14** | **RD—Read** | |
| | Read control bit. When set together with CSR9<12>, the 21041 performs read cycles from the boot ROM. | |
| **13** | **WR—Write** | |
| | Write control bit. When set together with CSR9<12>, the 21041 performs write cycles to the boot ROM (FLASH). | |
| **12** | **BR—Boot ROM Select** | |
| | When set, the 21041 selects the boot ROM. | |
| **11** | **SR—Serial ROM Select** | |
| | When set together with CSR9<13> or CSR9<14>, the 21041 selects the serial ROM. | |
| **10** | **REG—External Register Select** | |
| | When set, the 21041 selects an external register. Refer to Section 8.5. | |
| **7:0** | **DATA—Boot ROM Data or Serial ROM Control** | |
| | If boot ROM is selected (CSR7<12> is set), this field contains the data to be read from and written to the boot ROM. | |
| | If serial ROM is selected, bits <3:0> are connected to the serial ROM control pins as follows: | |
| | CSR9<3> | **Data Out**—This bit is used to serially shift the read data from the serial ROM device into the 21041. |
| | CSR9<2> | **Data In**—This bit is used to serially shift the write data from the 21041 into the serial ROM device. |
| | CSR9<1> | **Serial ROM Clock** —This bit is used as a serial ROM clock. |
| | CSR9<0> | **Serial ROM Chip Select**—This bit is used as a serial ROM chip select. |

Table 3–48 lists the access rules for the CSR9 register.

**Table 3–48 CSR9 Access Rules**

| Category | Description |
|---|---|
| Value after reset | FFFF03FFH |
| Read access rules | – |
| Write access rules | Writing to CSR9 bit 15 has no effect. |

### 3.3.11 Boot ROM Address Register (CSR10)

The CSR10 register contains the boot ROM address. Figure 3–20 shows the CSR10 bit field, and Table 3–49 describes the bit field.

**Figure 3–20 CSR10 Boot ROM Address Register**

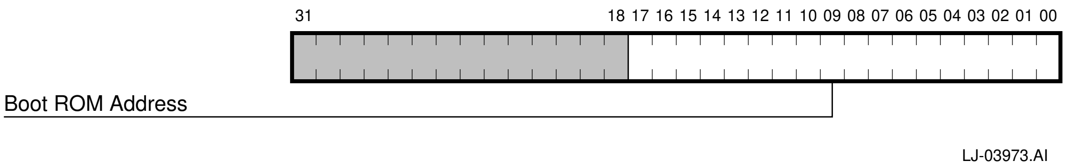



**Table 3–49 CSR10 Boot ROM Address Register Description**

| Field | Description |
|---|---|
| **17:0** | **Boot ROM Address**<br>Contains a pointer to the boot ROM. |

Table 3–50 lists the access rules for the CSR10 register.

**Table 3–50 CSR10 Access Rules**

| Category | Description |
|---|---|
| Value after reset | UNPREDICTABLE |
| Read access rules | – |
| Write access rules | – |

### 3.3.12 General-Purpose Timer Register (CSR11)

The CSR11 register contains a 16-bit timer for general purposes. It is used mainly by the software driver for timing functions that are not always supplied by the operating system.

Figure 3–21 shows the CSR11 bit fields, and Table 3–51 describes the bit fields.

**Figure 3–21 CSR11 General-Purpose Timer Register**

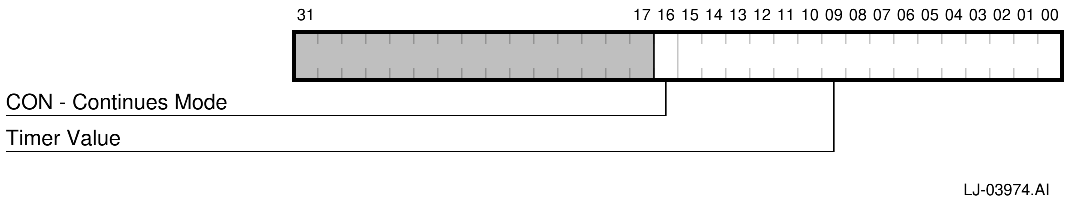



**Table 3–51 CSR11 General-Purpose Timer Register Description**

| Field | Description |
|---|---|
| **16** | **CON—Continues Mode**<br>When set, the timer works in continuous operating mode. When reset, the timer works in 1-shot operating mode. |
| **15:0** | **Timer Value**<br>Contains the timer value in 204.8-microsecond cycles. |

Table 3–52 lists the access rules for the CSR11 register.

**Table 3–52 CSR11 Access Rules**

| Category | Description |
|---|---|
| Value after reset | 00000000H |
| Read access rules | – |
| Write access rules | – |

### 3.3.13 Serial Interface Attachment CSRs

This section describes the four serial interface attachment (SIA) registers: CSR12, CSR13, CSR14, and CSR15. The description includes different SIA configurations and diagnostic programming. SIA status is maintained in CSR12.

The SIA registers control the functionality and connectivity of the SIA features, enabling various configurations and options. Some of the configurations are used only for diagnostic and testing purposes.

**Note**

Before changing any value in CSR13, CSR14, or CSR15, first perform an SIA software reset by writing CSR13 with all zeros (CSR13 = 00000000H). After writing the SIA registers, wait 10 milliseconds before writing CSR6 with start receive or transmit.

#### 3.3.13.1 SIA Status Register (CSR12)

The CSR12 register reads SIA pins and internal states. Figure 3–22 shows the CSR12 bit fields, and Table 3–53 describes the bit fields.

**Figure 3–22 CSR12 SIA Status Register**

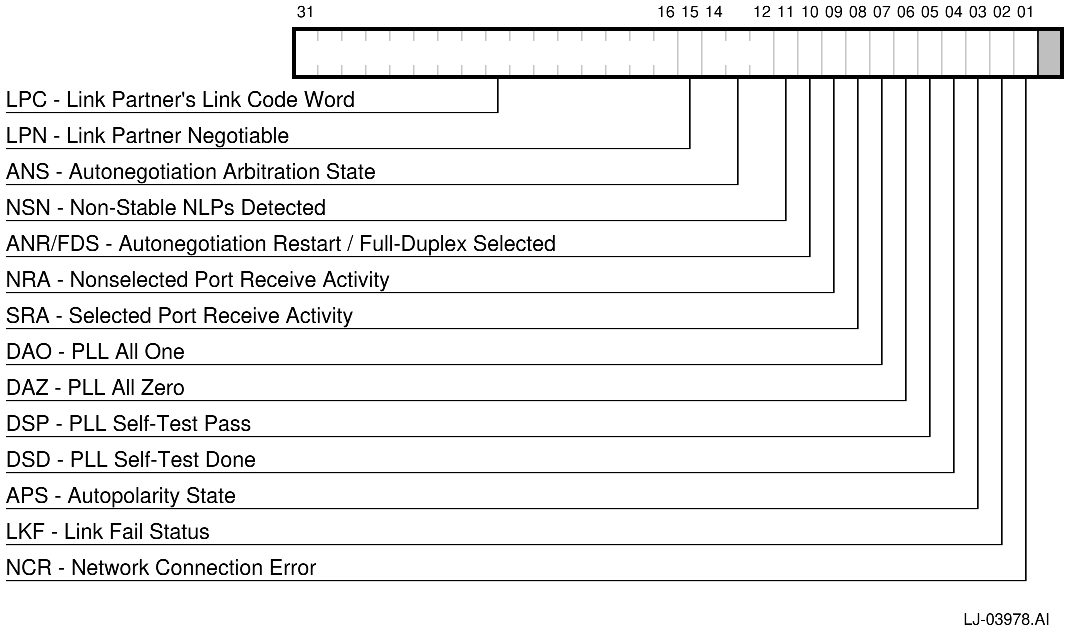


**Table 3–53 CSR12 SIA Status Register Description**

| Field | Description |
|---|---|
| **31:16** | **LPC—Link Partner’s Link Code Word**<br>These bits contain the link partner’s link code word, where bit 31 is S0 (selector field bit 0) and bit 16 is NP (Next Page). Meaningful only when CSR12<15> is read as a logical 1. |
| **15** | **LPN—Link Partner Negotiable**<br>This bit is set when the link partner is recognized to be a device that implements the autonegotiation algorithm. Meaningful only when CSR14<7> is set. |
| **14:12** | **ANS—Autonegotiation Arbitration State**<br>These bits reflect the current autonegotiation arbitration state as follows:<br>CSR12<14:12> = 000 - Autonegotiation disable<br>CSR12<14:12> = 001 - Transmit disable<br>CSR12<14:12> = 010 - Ability detect<br>CSR12<14:12> = 011 - Acknowledge detect<br>CSR12<14:12> = 100 - Complete acknowledge<br>CSR12<14:12> = 101 - FLP link good – autonegotiation complete<br>CSR12<14:12> = 110 - Link check<br>The autonegotiation is completed when a link pass interrupt is received and CSR12<14:12> are read as 101. |
| **11** | **NSN—Non-Stable NLPs Detected**<br>When set, indicates that the 10BASE-T normal link pulse (NLP) is not stable. The Link Integrity Test passed for a while, but failed later during negotiation. This means that NLPs were recognized on the line, but were not stable enough to cause autonegotiation completion.<br>This bit is cleared when a new autonegotiation sequence is started. Meaningful only when CSR14<7> is set. |
| **10** | **ANR/FDS—Autonegotiation Restart/Full-Duplex Selected**<br>When set by the driver, the autonegotiation sequence starts again (providing that CSR14<7> is set). At the end of the autonegotiation process, this bit indicates the operation mode agreed with the link partner. Reading 1 means the selected mode is full-duplex. Reading 0 means the selected mode is half-duplex. |
| **9** | **NRA—Nonselected Port Receive Activity**<br>This bit is set when there is receive activity on the nonselected port. |

(continued on next page)

**Table 3–53 (Cont.) CSR12 SIA Status Register Description**

| Field | Description |
|---|---|
| **8** | **SRA—Selected Port Receive Activity**<br>This bit is set when there is receive activity on the selected port. |
| **7** | **DAO—PLL All One**<br>Diagnostic bit. When set, indicates that all phase-locked loop (PLL) sampler synchronizers are asserted high. |
| **6** | **DAZ—PLL All Zero**<br>Diagnostic bit. When set, indicates that all PLL sampler synchronizers are asserted low. |
| **5** | **DSP—PLL Self-Test Pass**<br>PLL built-in integrity self-test status indicator (self-test start CSR15<12>).<br>PLL self-test pass (CSR12<5>) is valid only if PLL self-test done (CSR12<4>) is read as 1. If PLL self-test done (CSR12<4>) is 1 and PLL self-test pass (CSR12<5>) is 1, the self-test is successful; otherwise, the self-test fails. |
| **4** | **DSD—PLL Self-Test Done**<br>Reset when PLL self-test is initiated. Set after self-test completes. |
| **3** | **APS—Autopolarity State**<br>When set, the 10BASE-T polarity is positive. When reset, the 10BASE-T polarity is negative. The received bit stream is inverted by the receiver. (Refer to auto polarity enable CSR14<13> and set polarity plus CSR14<14>.) |
| **2** | **LKF—Link Fail Status**<br>This bit continuously reflects the 10BASE-T link test status. When set, the 10BASE-T link test is in fail state. When reset, the 10BASE-T link test is in pass state. This bit is meaningful only when CSR14<8>, Receive Squelch Enable, is set.<br>During link fail, the 21041 does not transmit any packet to the media. However, any queued packets in the transmit list can be closed by the 21041 with the following set:<br>TDES0<2>—Link fail<br>TDES0<10>—No carrier<br>TDES0<11>—Loss of carrier |

(continued on next page)

**Table 3–53 (Cont.) CSR12 SIA Status Register Description**

| Field | Description |
|---|---|
| | The 21041 moves from the link fail state to the link pass state when it receives a legal link pulse stream or two consecutive packets. The driver receives no indication about these packets. Following this, no transmit packet is pending and no carrier is sensed. |
| **1** | **NCR—Network Connection Error**<br>This bit has two meanings:<br>• In AUI, when set, it indicates no carrier. The status resets itself during the next transmission attempt.<br>• In 10BASE-T, this bit sets if no link pass state was established within 2.4 seconds from the beginning of the link test (indicating cable failure, for example) or after a link fail event occurred. If a link pass state was established, this bit resets. |

Table 3–54 lists the access rules for the CSR12 register.

**Table 3–54 CSR12 Access Rules**

| Category | Description |
|---|---|
| Value after reset | 000000C4H |
| Read access rules | – |
| Write access rules | CSR12<8> and CSR12<9> are cleared by writing 1. Writing 0 to these bits has no effect. Writing to the rest of the bits (except for bit 10) has no effect. |

### 3.3.13.2 SIA Connectivity Register (CSR13)

The CSR13 register contains the SIA connectivity control bits that permit the interconnection of different sections within the SIA. This allows coverage of the required operation and test options.

Figure 3–23 shows the CSR13 bit fields, and Table 3–55 describes the bit fields.

**Figure 3–23 CSR13 SIA Connectivity Register**

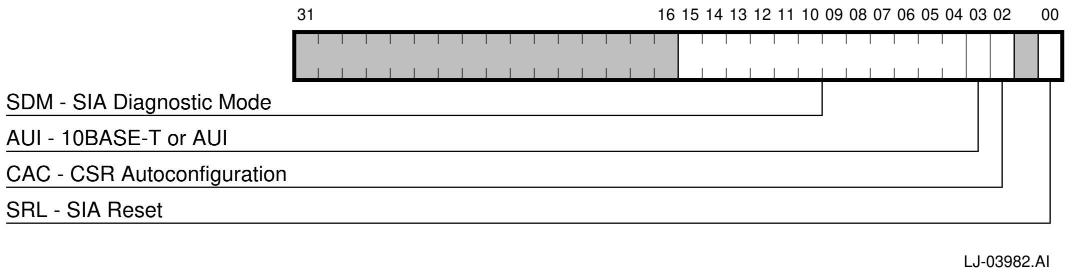



**Table 3–55 CSR13 SIA Connectivity Register Description**

| Field | Description |
|---|---|
| **15:4** | **SDM—SIA Diagnostic Mode** |
| | These bits are for engineering purposes only and must be set to EF0H. |
| **3** | **AUI—10BASE-T or AUI** |
| | When reset, forces the 21041 to select the 10BASE-T interface. When set to 1, forces the 21041 to select the AUI interface. |
| **2** | **CAC—CSR Autoconfiguration** |
| | When set, forces CSR13, CSR14, and CSR15 into a predetermined value according to the selection of the AUI CSR13<3> bit. |
| **0** | **SRL—SIA Reset** |
| | When reset, resets all the SIA functions and machines. |

Table 3–56 lists the access rules for the CSR13 register.

**Table 3–56 CSR13 Access Rules**

| Category | Description |
|---|---|
| Value after reset | FFFF0000H |
| Read access rules | If CSR auto configuration CSR13<2> is set, the value of CSR13 reflects the internal states rather than the values written into the CSR. |
| Write access rules | CSR13 should be reset to 00000000H before writing to any SIA CSR and released with or after the last CSR write. |

### 3.3.13.3 SIA Transmit and Receive Register (CSR14)

The CSR14 register configures the SIA transmitter and receiver operating modes. Figure 3–24 shows the CSR14 bit fields, and Table 3–57 describes the bit fields. This register is mainly used for diagnostic purposes.

**Figure 3–24 CSR14 SIA Transmit and Receive Register**

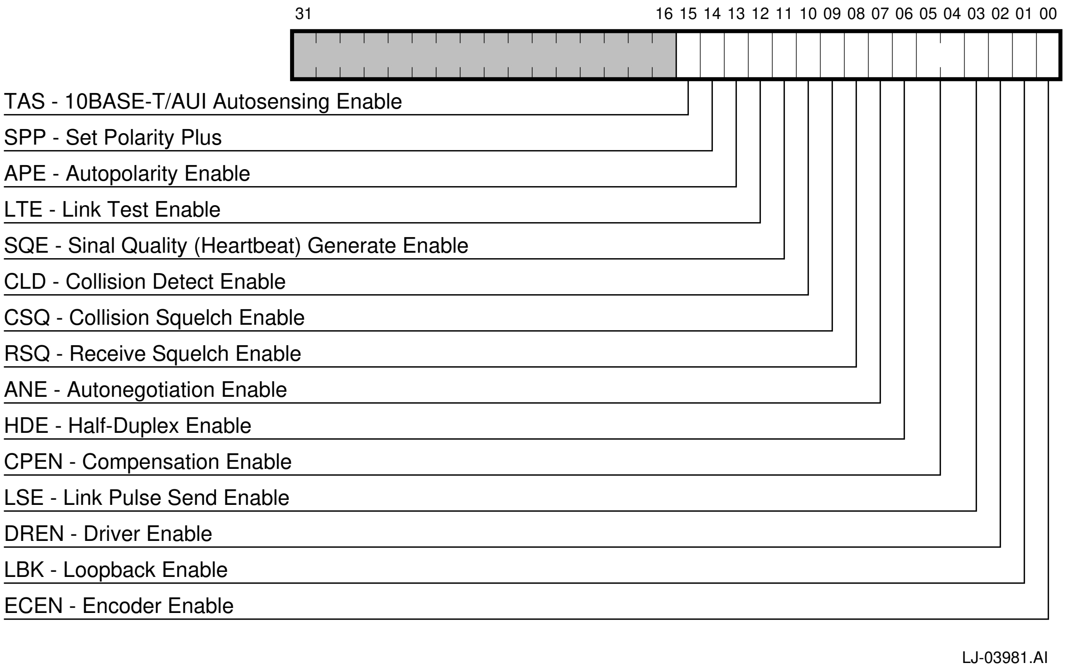


**Table 3–57 CSR14 SIA Transmit and Receive Register Description**

| Field | Description |
|---|---|
| **15** | **TAS—10BASE-T/AUI Autosensing Enable**<br>When set, the 21041 monitors its two ports (AUI and TP) in parallel for autosensing purposes. The selected port operation is not affected. See Section 7.1.8.<br>When cleared, the 21041 monitors only the port that is selected for operation (AUI or TP) according to CSR13<3>. |
| **14** | **SPP—Set Polarity Plus**<br>When reset and autopolarity enable (CSR14 <13>) is reset, the polarity of the incoming data is switched. This feature can be used by the driver to reverse polarity of incoming packets; otherwise, this bit should be set. This bit is valid only in 10BASE-T mode. |
| **13** | **APE—Autopolarity Enable**<br>When set and link test enable CSR14<12> is also set, the autopolarity function logic is enabled (Section 7.1.7). When reset, the polarity is determined by set polarity plus (CSR14<14>). When link test enable (CSR14<12>) is reset, this bit (CSR14<13>) should be also reset. This bit is valid only in 10BASE-T mode. |
| **12** | **LTE—Link Test Enable**<br>This bit is meaningful only for the 10BASE-T port. When set, the link test function logic is enabled. Resetting this bit forces the link test function to link pass state. |
| **11** | **SQE—Signal Quality (Heartbeat) Generate Enable**<br>Controls the signal quality (SQE) generator ability to imitate external medium attachment unit (MAU) behavior. When set, a short heartbeat signal is generated after the conclusion of a transmitted packet. In 10BASE-T mode, SQE (CSR14<11>) should be set; otherwise, a heartbeat fail (TDES0<7>) is set. In AUI mode, SQE (CSR14<11>) should be reset. |
| **10** | **CLD—Collision Detect Enable**<br>When set, the collision detect logic is enabled. |
| **9** | **CSQ—Collision Squelch Enable**<br>When set, the AUI collision receivers are active. This bit is valid only when AUI is selected. |
| **8** | **RSQ—Receive Squelch Enable**<br>When set, the AUI or 10BASE-T receivers are active in accordance with the selected mode. Note that when port autosensing is enabled, the AUI and 10BASE-T receivers are active simultaneously. |

(continued on next page)

**Table 3–57 (Cont.) CSR14 SIA Transmit and Receive Register Description**

| Field | Description |
|---|---|
| **7** | **ANE—Autonegotiation Enable**<br>This bit is meaningful only for the 10BASE-T port. When set, the 21041 performs a TP autonegotiation with the link partner to determine whether the network environment is half- or full-duplex.<br>When clear, autonegotiation is disabled. |
| **6** | **HDE—Half-Duplex Enable**<br>This bit controls the value of bit 5 in the transmitted Link Code Word.<br>When set, the 21041 advertises its ability to also work in half-duplex mode. (Bit 5 in the Link Code Word is set.)<br>When clear, the 21041 advertises that no half-duplex operation is allowed. (Bit 5 in the Link Code Word is cleared.)<br>This bit is meaningful only if CSR14<7> is set. |
| **5:4** | **CPEN—Compensation Enable**<br>Table 3–59 defines twisted-pair compensation behavior. These bits are valid only in 10BASE-T mode. |
| **3** | **LSE—Link Pulse Send Enable**<br>This bit is meaningful only for the 10BASE-T port. When set, the link pulse generator is enabled. |
| **2** | **DREN—Driver Enable**<br>When set, the transmit SIA driver is enabled for AUI or 10BASE-T operation. When reset, the transmit driver is disabled, preventing the data and link pulse transmission to the external wires. |
| **1** | **LBK—Loopback Enable**<br>Enables loopback operation in SIA (Table 3–62 and Section 5.7.3). In AUI mode, this bit should be reset. |
| **0** | **ECEN—Encoder Enable**<br>When set, the transmit data encoder is enabled, and the encoded data is transferred to the output drivers. When reset, the transmit data encoder is disabled, and the encoded data is blocked from propagating to the output drivers. |

Table 3–58 lists the access rules for the CSR14 register.

**Table 3–58 CSR14 Access Rules**

| Category | Description |
|---|---|
| Value after reset | FFFFFFFFH |
| Read access rules | In SIA_auto_configuration mode, a CSR14 read reflects internal states, rather than the values written into the CSR. |
| Write access rules | CSR13 should be reset to 00000000H before writing any SIA CSR and released with or just after the last CSR write. |

Table 3–59 lists the compensation field (CSR14<5:4>) definitions.

**Table 3–59 Twisted-Pair Compensation Behavior**

| CSR14<5:4> Value | Transmitter Output |
|---|---|
| 00, 01 | Compensation Disabled Mode—Twisted-pair driver does not compensate for 10-megahertz versus 5-megahertz media attenuation. (Differential voltages are bound between 1.5 volts and 2.1 volts.) |
| 10 | High Power Mode—Twisted-pair driver drives only high-differential voltage (between 2.2 volts and 2.8 volts). |
| 11 | Normal Compensation Mode—Driver compensates for 10-megahertz versus 5-megahertz media attenuation by driving high-differential voltage for transients and driving low if the signal is stable for more than 50 nanoseconds. |

#### 3.3.13.4 SIA General Register (CSR15)

Figure 3–25 shows the CSR15 bit fields, and Table 3–60 describes the bit fields. This register is mainly used for diagnostic purposes.

**Figure 3–25 CSR15 SIA General Register**

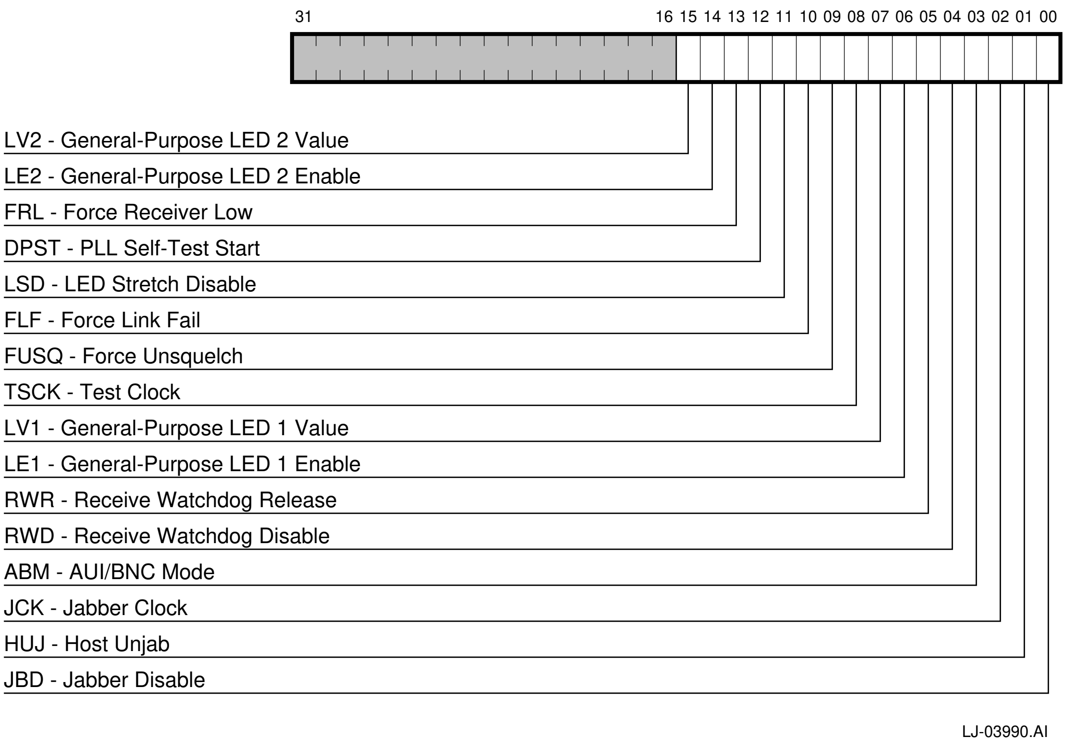


**Table 3–60 CSR15 SIA General Register Description**

| Field | Description |
|---|---|
| **15** | **LV2—General-Purpose LED 2 Value**<br>Meaningful only when CSR15<14> is set. When set, the General-Purpose LED 2 (pin 117) is on. When clear, the General-Purpose LED 2 (pin 117) is off. |
| **14** | **LE2—General-Purpose LED 2 Enable**<br>When set, pin 117 is used as a general-purpose LED. When clear, pin 117 is used as a TP Polarity LED. |
| **13** | **FRL—Force Receiver Low**<br>Testing feature that forces the RX input to PLL samplers to a constant low. It is used to detect stuck samplers. |
| **12** | **DPST—PLL Self-Test Start**<br>Testing feature that starts the PLL built-in integrity self-test. The status of the test result is marked by PLL self-test done (CSR12<4>) and PLL self-test pass (CSR12<5>) respectively. |
| **11** | **LSD—LED Stretch Disable**<br>When set, the LED stretcher is disabled and the internal signal level is directly reflected on the LED pins. When clear, the LED stretcher is active, making each LED setting last for at least 52 milliseconds. |
| **10** | **FLF—Force Link Fail**<br>Testing feature that forces a link fail state and resets both link test and autopolarity detector. This bit is valid only when CSR14<12>, Link Test Enable, is set. |
| **9** | **FUSQ—Force Unsquelch**<br>Testing feature that asserts the receiver RCVEN signal for testing purposes. |
| **8** | **TSCK—Test Clock**<br>Testing feature that increases the clock frequency of certain SIA clocks. |
| **7** | **LV1—General-Purpose LED 1 Value**<br>Meaningful only when CSR15<6> is set. When set, the General-Purpose LED 1 (pin 118) is on. When clear, the General-Purpose LED 1 (pin 118) is off. |

(continued on next page)

**Table 3–60 (Cont.) CSR15 SIA General Register Description**

| Field | Description |
|---|---|
| **6** | **LE1—General-Purpose LED 1 Enable**<br>When set, pin 118 is used as a general-purpose LED. When clear, pin 118 is used as a Transmit Jabber LED. |
| **5** | **RWR—Receive Watchdog Release**<br>Defines the time interval *no carrier* from receive watchdog expiration until re-enabling the receive channel. When set, the receive watchdog is released 40- to 48-bit-times from the last carrier deassertion. When reset, the receive watchdog is released 16- to 24-bit-times from the last carrier deassertion. |
| **4** | **RWD—Receive Watchdog Disable**<br>When set, the receive watchdog counter is disabled. When clear, receive carriers longer than 2560 bytes are guaranteed to cause the watchdog counter to timeout. Packets shorter than 2048 bytes are guaranteed to pass. |
| **3** | **ABM—AUI/BNC Mode**<br>This bit is used by the driver to select AUI or BNC mode. When set, 10BASE5 (AUI) is selected. When clear, 10BASE2 (BNC) is selected. |
| **2** | **JCK—Jabber Clock**<br>When set, transmission is cut after 2048 to 2560 bytes are transmitted (1.6 to 2.0 milliseconds). When reset, transmission is cut after 26 to 33 milliseconds. |
| **1** | **HUJ—Host Unjab**<br>Defines the time interval between transmit jabber expiration until re-enabling of the transmit channel. When set, the transmit channel is released immediately after the jabber expiration. When reset, the transmit jabber is released 365 to 420 milliseconds after jabber expiration. |
| **0** | **JBD—Jabber Disable**<br>When set, the transmit jabber function is disabled. |

Table 3–61 lists the access rules for the CSR15 register.

**Table 3–61 CSR15 Access Rules**

| Category | Description |
|---|---|
| Value after reset | FFFF0000H |
| Read access rules | A CSR15 read reflects internal states, rather than the values written into the CSR. |
| Write access rules | CSR13 should be reset to 00000000H before writing any SIA CSR and released with or just after the last CSR write. |

#### 3.3.13.5 SIA Operational Modes

Table 3–62 lists the programming of the different SIA modes using CSR13, CSR14, and CSR15. The states of operating mode CSR6<11:10> and full-duplex mode CSR6<9> are also identified.

**Table 3–62 Programming of SIA Modes Using CSR13, CSR14, and CSR15**

| Mode | CSR13 | CSR14 | CSR15 | AUI_BNC Pin | CSR6<FD> | CSR6<OM> |
|---|---|---|---|---|---|---|
| **Autosensing Disabled, Autonegotiation Disabled** | | | | | | |
| 10BASE-T forced to half-duplex | EF01 | 7F3F | 0008 | high (AUI) | 0 | 00 |
| 10BASE-T forced to full-duplex | EF01 | 7F3D | 0008 | high (AUI) | 1 | 00 |
| 10BASE-T internal loopback | EF01 | 7A3F | 0008 | high (AUI) | 0 | 10 |
| 10BASE-T external loopback | EF01 | 7B3D | 0008 | high (AUI) | 0 | 10 |
| BNC (10BASE2) | EF09 | 0705 | 0006 | low (BNC) | 0 | 00 |
| BNC (10BASE2) external loopback | EF09 | 0705 | 0006 | low (BNC) | 0 | 10 |
| AUI (10BASE5) | EF09 | 0705 | 000E | high (AUI) | 0 | 00 |
| AUI (10BASE5) external loopback | EF09 | 0705 | 000E | high (AUI) | 0 | 10 |
| Internal loopback in MAC level | EF09 | 0000 | 0019 | high (AUI) | 0 | 01 |
| **Autosensing Disabled, Autonegotiation Enabled** | | | | | | |
| 10BASE-T advertising half- and full-duplex | EF01 | 7FFF | 0008 | high (AUI) | 1 | 00 |
| 10BASE-T advertising full-duplex | EF01 | 7FBF | 0008 | high (AUI) | 1 | 00 |
| 10BASE-T advertising half-duplex | EF01 | 7FFF | 0008 | high (AUI) | 0 | 00 |

(continued on next page)

**Table 3–62 (Cont.) Programming of SIA Modes Using CSR13, CSR14, and CSR15**

| Mode | CSR13 | CSR14 | CSR15 | AUI_BNC Pin | CSR6<FD> | CSR6<OM> |
|---|---|---|---|---|---|---|
| **Autosensing Enabled, Autonegotiation Disabled** | | | | | | |
| 10BASE-T forced to half-duplex | EF01 | FF3F | 0008 | high (AUI) | 0 | 00 |
| 10BASE-T forced to full-duplex | EF01 | FF3D | 0008 | high (AUI) | 1 | 00 |
| BNC (10BASE2) | EF09 | F73D | 0006 | low (BNC) | 0 | 00 |
| AUI (10BASE5) | EF09 | F73D | 000E | high (AUI) | 0 | 00 |
| **Autosensing Enabled, Autonegotiation Enabled** | | | | | | |
| 10BASE-T advertising half- and full-duplex | EF01 | FFFF | 0008 | high (AUI) | 1 | 00 |
| 10BASE-T advertising full-duplex | EF01 | FFBF | 0008 | high (AUI) | 1 | 00 |
| 10BASE-T advertising half-duplex | EF01 | FFFF | 0008 | high (AUI) | 0 | 00 |
| BNC (10BASE2) advertising half- and full-duplex on TP | EF09 | F7FD | 0006 | low (BNC) | 1 | 00 |
| BNC (10BASE2) advertising full-duplex only on TP | EF09 | F7BD | 0006 | low (BNC) | 1 | 00 |
| BNC (10BASE2) advertising half-duplex only on TP | EF09 | F7FD | 0006 | low (BNC) | 0 | 00 |

(continued on next page)

**Table 3–62 (Cont.) Programming of SIA Modes Using CSR13, CSR14, and CSR15**

| Mode | CSR13 | CSR14 | CSR15 | AUI_BNC Pin | CSR6<FD> | CSR6<OM> |
|---|---|---|---|---|---|---|
| **Autosensing Enabled, Autonegotiation Enabled** | | | | | | |
| AUI (10BASE5) advertising half- and full-duplex on TP | EF09 | F7FD | 000E | high (AUI) | 1 | 00 |
| AUI (10BASE5) advertising full-duplex only on TP | EF09 | F7BD | 000E | high (AUI) | 1 | 00 |
| AUI (10BASE5) advertising half-duplex only on TP | EF09 | F7FD | 000E | high (AUI) | 0 | 00 |

# 4

# Host Communication Area

This chapter describes the descriptors and data buffers, which are collectively called the host communication area. They reside in the host memory and manage the actions and status related to buffer management.

## 4.1 Data Communication

The 21041 and the driver communicate through two data structures:

- **Command and status registers** (CSRs) are described in Chapter 3.
- **Descriptor lists and data buffers** are described in this chapter.

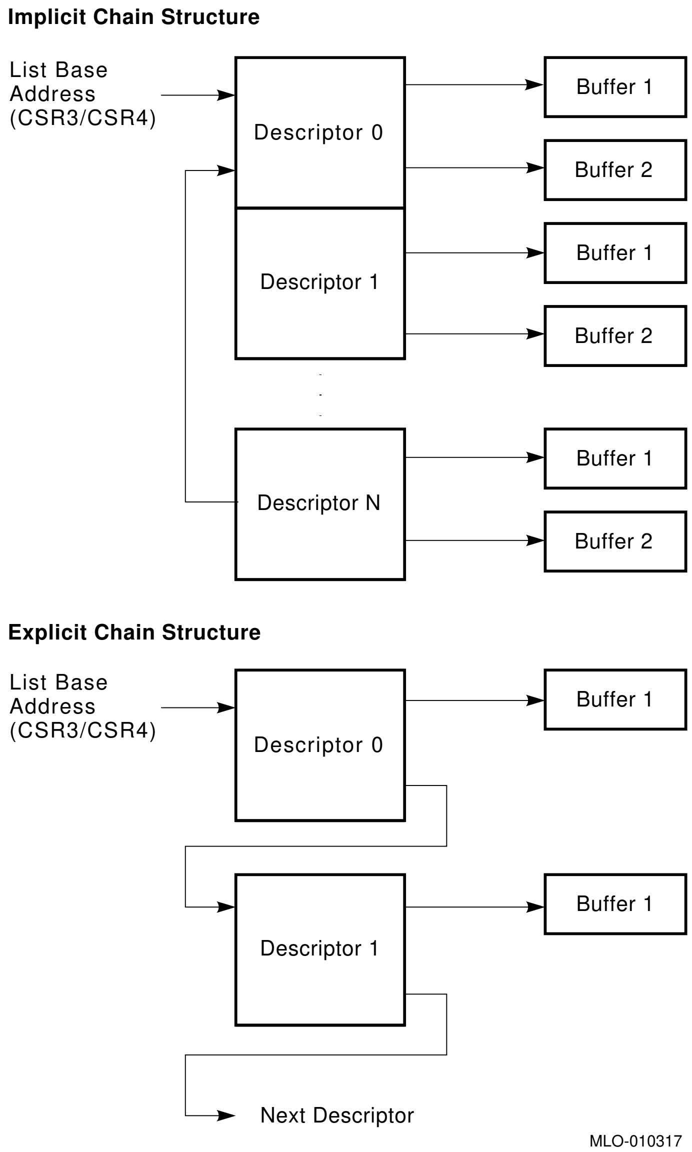
## 4.2 Descriptor Lists and Data Buffers

The 21041 transfers frame data to and from the receive and transmit buffers in host memory. Descriptors that reside in the host memory act as pointers to these buffers. The descriptor lists are mapped in the host *physical* memory address space.

There are two descriptor lists, one for receive and one for transmit. The base address of each list is written into CSR3 and CSR4, respectively. A descriptor list is forward-linked either implicitly or explicitly. Explicit chaining of descriptors is accomplished by setting the Second Address Chained bit in both the receive and transmit descriptors (RDES1<24> and TDES1<24>). In this case, the pointer to the next descriptor resides in the Buffer Address 2 field (RDES3 and TDES3). The last descriptor may point back to the first entry to create a ring structure.

Implicit chaining of descriptors is accomplished by writing the Second Address Chained bit (RDES1<24> and TDES1<24>) as 0. In this case, each descriptor can point to a maximum of two buffers. This enables two buffers to be used, physically addressed, and not contiguous in memory. In the last descriptor, the End of Ring bit (RDES1<25> and TDES1<25>) should be set, thus creating a ring structure. Figure 4–1 shows an example of implicit and explicit descriptors chaining.

**Figure 4–1 Descriptor Implicit and Explicit Chain Structures**



A data buffer consists of either an entire frame or part of a frame, but it cannot exceed a single frame. Buffers contain only data; buffer status is maintained in the descriptor. Data chaining refers to frames that span multiple data buffers. Data chaining can be enabled or disabled. Data buffers reside in host *physical* memory space.

### 4.2.1 Receive Descriptors

Figure 4–2 shows the receive descriptor format.

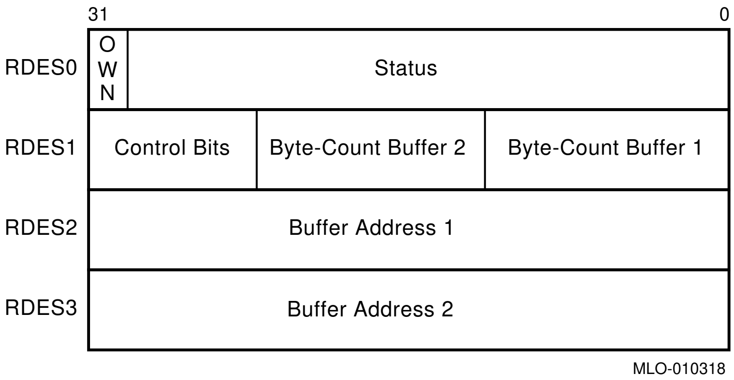
**Note**

Descriptors and receive buffer addresses must be longword-aligned.

---

Providing two buffers, two byte-count buffers, and two address pointers in each descriptor enables the adapter port to be compatible with various types of memory-management schemes.

**Figure 4–2 Receive Descriptor Format**




#### 4.2.1.1 Receive Descriptor 0 (RDES0)

RDES0 contains the received frame status, the frame length, and the descriptor ownership information. Figure 4–3 shows the RDES0 bit fields, and Table 4–1 describes the bit fields.

**Figure 4–3 RDES0 Receive Descriptor 0**

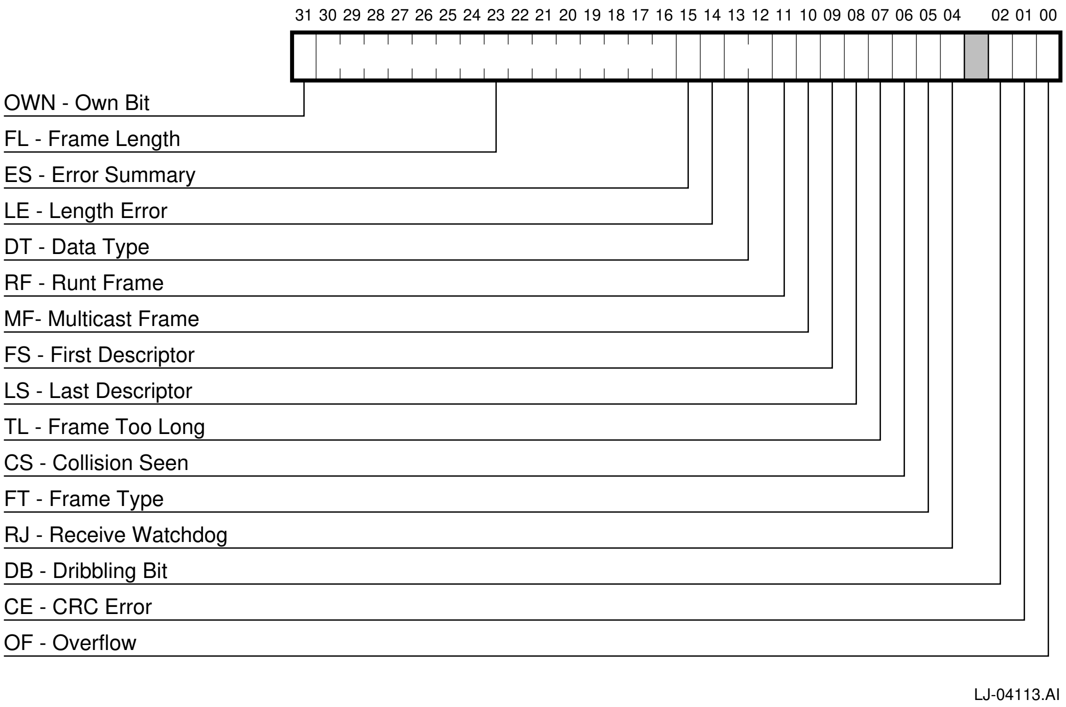


**Table 4–1 RDES0 Receive Descriptor 0 Description**

| Field | Description |
|---|---|
| **31** | **OWN—Own Bit**<br><br>When set, indicates that the descriptor is owned by the 21041. When reset, indicates that the descriptor is owned by the host. The 21041 clears this bit either when it completes the frame reception or when the buffers that are associated with this descriptor are full. |
| **30:16** | **FL—Frame Length**<br><br>Indicates the length in bytes of the received frame including the cyclic redundancy check (CRC).<br><br>This field is valid only when last descriptor (RDES0<8>) is set and length error (RDES0<14>) is reset. |
| **15** | **ES—Error Summary**<br><br>Indicates the logical OR of the following RDES0 bits:<br><br>RDES0<0>—Overflow<br>RDES0<1>—CRC error<br>RDES0<6>—Collision seen<br>RDES0<7>—Frame too long<br>RDES0<11>—Runt frame<br>RDES0<14>—Length error<br><br>This field is valid only when last descriptor (RDES0<8>) is set. |
| **14** | **LE—Length Error**<br><br>When set, indicates a frame truncation caused by a frame that does not fit within the current descriptor buffers and indicates that the 21041 does not own the next descriptor. The frame is truncated.<br><br>This field is valid only when last descriptor (RDES0<8>) is set. |
| **13:12** | **DT—Data Type**<br><br>This field is valid only when last descriptor (RDES0<8>) is set. |

(continued on next page)

**Table 4–1 (Cont.) RDES0 Receive Descriptor 0 Description**

| Field | Description |
|---|---|
| | Indicates the type of frame the buffer contains.<br>00—Serial received frame.<br>01—Internal loopback frame.<br>10—External loopback frame or serial received frame. The 21041 does not differentiate between loopback and serial received frames; therefore, this information is global and reflects only the operating mode (CSR6<11:10>).<br>11—Reserved. |
| **11** | **RF—Runt Frame**<br>When set, indicates that the frame was damaged by a collision or premature termination before the collision window had passed. Runt frames are passed on to the host only if the pass bad frames bit (CSR6<3>) is set.<br>This field is valid only when last descriptor (RDES0<8>) is set and overflow (RDES0<0>) is reset. |
| **10** | **MF—Multicast Frame**<br>When set, indicates that this frame has a multicast address.<br>This field is valid only when last descriptor (RDES0<8>) is set. |
| **9** | **FS—First Descriptor**<br>When set, indicates that this descriptor contains the first buffer of a frame.<br>If the buffer size of the first buffer is 0, the second buffer contains the beginning of the frame. If the buffer size of the second buffer is also 0, the second descriptor contains the beginning of the frame. |
| **8** | **LS—Last Descriptor**<br>When set, indicates that the buffers pointed to by this descriptor, are the last buffers of the frame. |
| **7** | **TL—Frame Too Long**<br>When set, indicates that the frame length exceeds the maximum Ethernet specified size of 1518 bytes.<br>This field is valid only when last descriptor (RDES0<8>) is set.<br>**Note**<br>Frame too long is only a frame length indication and does not cause any frame truncation. |

(continued on next page)

**Table 4–1 (Cont.) RDES0 Receive Descriptor 0 Description**

| Field | Description |
|---|---|
| **6** | **CS—Collision Seen**<br>When set, indicates that the frame was damaged by a collision that occurred after the 64 bytes following the start frame delimiter (SFD). This is a late collision.<br>This field is valid only when last descriptor (RDES0<8>) is set. |
| **5** | **FT—Frame Type**<br>When set, indicates that the frame is an Ethernet type frame (frame length field is greater than 1500 bytes). When clear, indicates that the frame is an IEEE 802.3 frame.<br>This field is not valid for runt frames of less than 14 bytes.<br>This field is valid only when last descriptor (RDES0<8>) is set. |
| **4** | **RJ—Receive Watchdog**<br>When set, indicates that the receive watchdog timer expired while receiving the current packet with length greater than 2048–2560 bytes. Receive watchdog timeout (CSR5<9>) is set.<br>This field is valid only when last descriptor (RDES0<8>) is set. |
| **2** | **DB—Dribbling Bit**<br>When set, indicates that the frame contained a non-integer multiple of 8 bits. This error is reported only if the number of dribbling bits in the last byte is greater than 2. This field is not valid if either collision seen (RDES0<6>) or runt frame RDES0<11> are set.<br>This field is valid only when last descriptor (RDES0<8>) is set. |
| **1** | **CE—CRC Error**<br>When set, indicates that a cyclic redundancy check (CRC) error occurred on the received frame.<br>The CRC check is performed independent of a dribbling bit (RDES0<2>) error. However, only whole bytes are run through the CRC logic. Consequently, received frames with up to 6 dribbling bits cause this bit to be set.<br>This field is valid only when last descriptor (RDES0<8>) is set. |
| **0** | **OF—Overflow**<br>When set, indicates received data in this descriptor's buffer were truncated due to FIFO overflow. This generally occurs if 21041 bus requests are not granted before the internal receive FIFO fills up.<br>This field is valid only when last descriptor (RDES0<8>) is set. |

#### 4.2.1.2 Receive Descriptor 1 (RDES1)

Figure 4–4 shows the RDES1 bit fields, and Table 4–2 describes the bit fields.

**Figure 4–4 RDES1 Receive Descriptor 1**

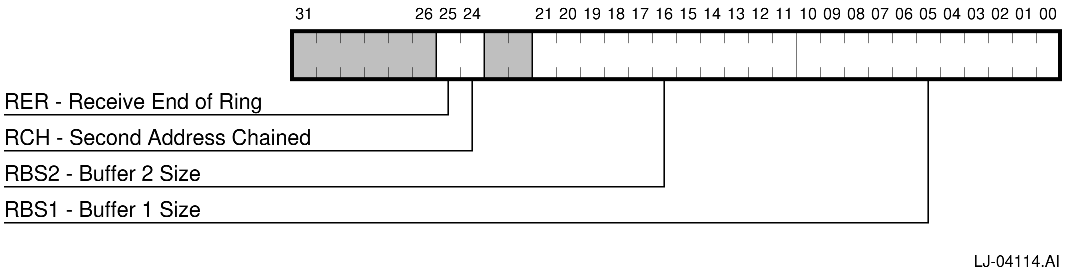



**Table 4–2 RDES1 Receive Descriptor 1 Description**

| Field | Description |
|---|---|
| **25** | **RER—Receive End of Ring** |
| | When set, indicates that the descriptor list reached its final descriptor. The 21041 returns to the base address of the list (Section 3.3.5) creating a descriptor ring. |
| **24** | **RCH—Second Address Chained** |
| | When set, indicates that the second address in the descriptor is the next descriptor address, rather than the second buffer address. |
| | RDES1<25> takes precedence over RDES1<24>. |
| **21:11** | **RBS2—Buffer 2 Size** |
| | Indicates the size, in bytes, of the second data buffer. If this field is 0, the 21041 ignores this buffer and fetches the next descriptor. |
| | The buffer size must be a multiple of 4. |
| | This field is not valid if RDES1<24> is set. |
| **10:0** | **RBS1—Buffer 1 Size** |
| | Indicates the size, in bytes, of the first data buffer. If this field is 0, the 21041 ignores this buffer and uses buffer 2. |
| | The buffer size must be a multiple of 4. |

### 4.2.1.3 Receive Descriptor 2 (RDES2)

Figure 4–5 shows the RDES2 bit field, and Table 4–3 describes the bit field.

**Figure 4–5 RDES2 Receive Descriptor 2**

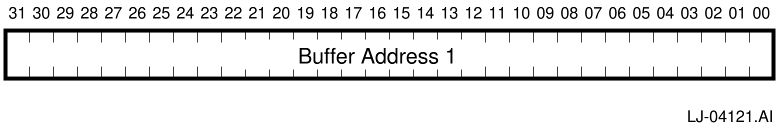



**Table 4–3 RDES2 Receive Descriptor 2 Description**

| Field | Description |
|---|---|
| **31:0** | **Buffer Address 1**<br>Indicates the physical address of buffer 1. The buffer must be longword-aligned (RDES2<1:0> = 00). |

### 4.2.1.4 Receive Descriptor 3 (RDES3)

Figure 4–6 shows the RDES3 bit field, and Table 4–4 describes the bit field.

**Figure 4–6 RDES3 Receive Descriptor 3**

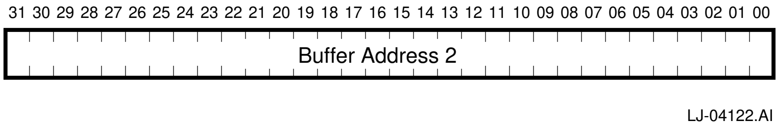



**Table 4–4 RDES3 Receive Descriptor 3 Description**

| Field | Description |
|---|---|
| **31:0** | **Buffer Address 2**<br>Indicates the physical address of buffer 2. The buffer must be longword-aligned (RDES3<1:0> = 00). |

#### 4.2.1.5 Receive Descriptor Status Validity

Table 4–5 lists the validity of the receive descriptor status bits in relation to the reception completion status. The following list defines the abbreviations used in the table:

RF—Runt frame (RDES0<11>)
CS—Collision seen (RDES0<6>)
FT—Frame type (RDES0<5>)
DB—Dribbling bit (RDES0<2>)
CE—CRC error (RDES0<1>)
ES—Error summary (RDES0<15>)
LE—Length error (RDES0<14>)
DT—Data type (RDES0<13:12>)
FS—First descriptor (RDES0<9>)
LS—Last descriptor (RDES0<8>)
FL—Frame length (RDES0<30:16>)
OF—Overflow (RDES0<0>)

**Table 4–5 Receive Descriptor Status Validity**

| | Receive Status Report | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Reception Status** | **RF** | **CS** | **FT** | **DB** | **CE** | **LE** | **FL** | **(ES, DT, FS, LS, OF)** |
| Overflow | NV | NV | V | NV | NV | V | V | V |
| Collision after 512 bits | V | V | V | NV | NV | V | V | V |
| Runt frame | V | V | V | NV | NV | V | V | V |
| Runt frame less than 14 bytes | V | V | NV | NV | NV | V | V | V |
| Watchdog timeout | V | NV | V | NV | NV | NV | NV | V |

V—Valid
NV—Not valid

## 4.2.2 Transmit Descriptors

Figure 4–7 shows the transmit descriptor format.

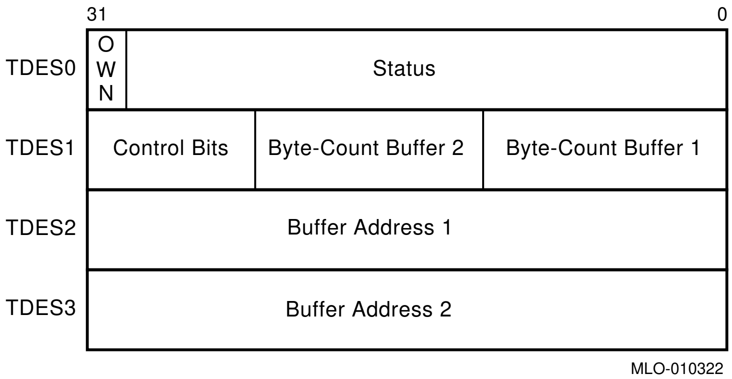
**Note**

Descriptor addresses must be longword-aligned.

---

Providing two buffers, two byte-count buffers, and two address pointers in each descriptor enables the adapter port to be compatible with various types of memory-management schemes.

**Figure 4–7 Transmit Descriptor Format**




### 4.2.2.1 Transmit Descriptor 0 (TDES0)

TDES0 contains transmitted frame status and descriptor ownership information. Figure 4–8 shows the TDES0 bit fields, and Table 4–6 describes the bit fields.

**Figure 4–8 TDES0 Transmit Descriptor 0**

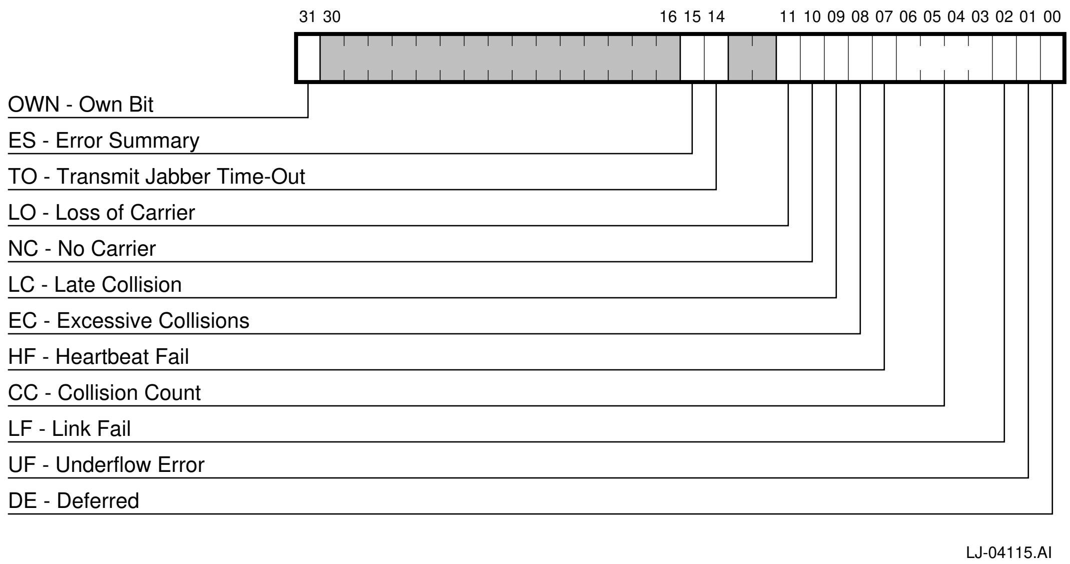


**Table 4–6 TDES0 Transmit Descriptor 0 Description**

| Field | Description |
|---|---|
| 31 | **OWN—Own Bit**<br>When set, indicates that the descriptor is owned by the 21041. When cleared, indicates that the descriptor is owned by the host. The 21041 clears this bit either when it completes the frame transmission or when the buffers that are allocated in the descriptor are empty.<br>The ownership bit of the first descriptor of the frame should be set after all subsequent descriptors belonging to the same frame have been set. This avoids a possible race condition between the 21041 fetching a descriptor and the driver setting an ownership bit. |
| 15 | **ES—Error Summary**<br>Indicates the logical OR of the following bits:<br>TDES0<1>—Underflow error<br>TDES0<2>—Link fail<br>TDES0<8>—Excessive collisions<br>TDES0<9>—Late collision<br>TDES0<10>—No carrier<br>TDES0<11>—Loss of carrier<br>TDES0<14>—Transmit jabber timeout |
| 14 | **TO—Transmit Jabber Timeout**<br>When set, indicates that the transmit jabber timer timed out and that the 21041 transmitter was babbling. The transmit jabber timeout interrupt CSR5<3> is set. The transmission process is aborted and placed in the stopped state. |
| 11 | **LO—Loss of Carrier**<br>When set, indicates loss of carrier during transmission (possible short circuit in the Ethernet cable).<br>Not valid in internal loopback mode (CSR6<11:10>=01). |
| 10 | **NC—No Carrier**<br>When set, indicates that the carrier signal from the transceiver was not present during transmission (possible problem in the transceiver or the transceiver cable).<br>Not valid in internal loopback mode (CSR6<11:10>=01). |
| 9 | **LC—Late Collision**<br>When set, indicates that the frame transmission was aborted due to collision occurring after the collision window of 64 bytes. Not valid if underflow error (TDES0<1>) is set. |

(continued on next page)

**Table 4–6 (Cont.) TDES0 Transmit Descriptor 0 Description**

| Field | Description |
|---|---|
| **8** | **EC—Excessive Collisions**<br>When set, indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current frame. |
| **7** | **HF—Heartbeat Fail**<br>When set, indicates a heartbeat collision check failure. (The transceiver failed to return a collision pulse as a check after the transmission.) Some transceivers do not support heartbeat; in this case, heartbeat fail is always set but not valid.<br>This bit is not valid if underflow error (TDES0<1>) is set.<br>In 10BASE-T mode, if signal quality enable (CSR14<11>) is reset, heartbeat fail (TDES0<7>) is always set.<br>In AUI mode or external SIA mode, signal quality enable (CSR14<11>) has no effect on heartbeat fail (TDES0<7>).<br>On the second transmission attempt, after the first transmission was aborted due to collision, the 21041 does not check heartbeat fail and (TDES0<7>) is reset. |
| **6:3** | **CC—Collision Count**<br>This 4-bit counter indicates the number of collisions that occurred before the frame was transmitted.<br>Not valid when the excessive collisions bit (TDES0<8>) is also set. |
| **2** | **LF—Link Fail**<br>When set, indicates that the link fail test failed and the twisted-pair line cannot transmit the packet. (See link fail status CSR12<2>.) |
| **1** | **UF—Underflow Error**<br>When set, indicates that the transmitter aborted the message because data arrived late from memory. Underflow error indicates that the 21041 encountered an empty transmit FIFO while transmitting a frame. The transmission process enters the suspended state and sets both transmit underflow (CSR5<5>) and transmit interrupt (CSR5<0>). |
| **0** | **DE—Deferred**<br>When set, indicates that the 21041 had to defer while ready to transmit a frame because the carrier was asserted. |

#### 4.2.2.2 Transmit Descriptor 1 (TDES1)

Figure 4–9 shows the TDES1 bit fields, and Table 4–7 describes the bit fields.

**Figure 4–9 TDES1 Transmit Descriptor 1**

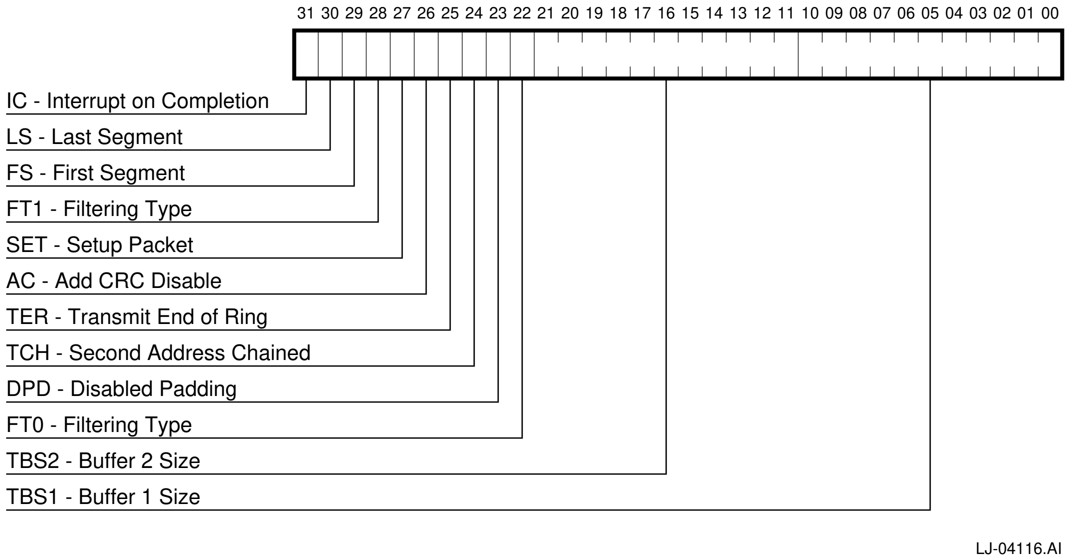



**Table 4–7 TDES1 Transmit Descriptor 1 Description**

| Field | Description |
|---|---|
| **31** | **IC—Interrupt on Completion** |
| | When set, the 21041 sets transmit interrupt (CSR5<0>) after the present frame has been transmitted. It is valid only when last segment (TDES1<30>) is set. |
| **30** | **LS—Last Segment** |
| | When set, indicates that the buffer contains the last segment of a frame. |
| **29** | **FS—First Segment** |
| | When set, indicates that the buffer contains the first segment of a frame. |
| **28** | **FT1—Filtering Type** |
| | Table 4–8 lists the filtering types. |

(continued on next page)

**Table 4–7 (Cont.) TDES1 Transmit Descriptor 1 Description**

| Field | Description |
|---|---|
| **27** | **SET—Setup Packet**<br>When set, indicates that the current descriptor is a setup frame descriptor (Section 4.2.3). |
| **26** | **AC—Add CRC Disable**<br>When set, the 21041 does not append the cyclic redundancy check (CRC) to the end of the transmitted frame. This field is valid only when first segment (TDES1<29>) is set. |
| **25** | **TER—Transmit End of Ring**<br>When set, indicates that the descriptor pointer has reached its final descriptor. The 21041 returns to the root address of the list (Section 3.3.5). This creates a descriptor ring. |
| **24** | **TCH—Second Address Chained**<br>When set, indicates that the second address in the descriptor is the next descriptor address, rather than the second buffer address.<br>Transmit end of ring (TDES1<25>) takes precedence over second address chained (TDES1<24>). |
| **23** | **DPD—Disabled Padding**<br>When set, the 21041 does not automatically add a padding field to a packet shorter than 64 bytes.<br>When reset, the 21041 automatically adds a padding field and a CRC field to a packet shorter than 64 bytes. The CRC field is added despite the state of the add CRC disable (TDES1<26>) flag. |
| **22** | **FT0—Filtering Type**<br>Table 4–8 lists the filtering types. |
| **21:11** | **TBS2—Buffer 2 Size**<br>Indicates the size, in bytes, of the second data buffer. If this field is 0, the 21041 ignores this buffer and fetches the next descriptor.<br>This field is not valid if second address chained (TDES1<24>) is set. |
| **10:0** | **TBS1—Buffer 1 Size**<br>Indicates the size, in bytes, of the first data buffer. If this field is 0, the 21041 ignores this buffer and uses buffer 2. |

Table 4–8 lists the filtering types. Section 3.3.7 provides additional information about filtering.

**Table 4–8 Filtering Types**

| FT1 | FT0 | Description |
|---|---|---|
| 0 | 0 | **Perfect filtering**<br>The 21041 interprets the descriptor buffer as a setup perfect table of 16 addresses and sets the 21041 filtering mode to perfect filtering.<br>This field is valid only when setup packet (TDES1<27>) is set (Table 3–40). |
| 0 | 1 | **Hash filtering**<br>The 21041 interprets the descriptor buffer as a setup hash table of 512-bit-plus-one perfect address. If an incoming receive packet destination address is a multicast address, the 21041 executes an imperfect address filtering compared with the hash table. However, if the incoming receive packet destination address is a physical address, the 21041 executes a perfect filtering compared with the perfect address.<br>This field is valid only when setup packet (TDES1<27>) is set (Table 3–40). |
| 1 | 0 | **Inverse filtering**<br>The 21041 interprets the descriptor buffer as a setup perfect table of 16 addresses and sets the 21041 filtering mode to inverse filtering.<br>The 21041 receives the incoming frames with destination addresses that do not match the perfect addresses and rejects the frames with destination addresses that match one of the perfect addresses.<br>This field is valid only when setup packet (TDES1<27>) is set (Table 3–40). |
| 1 | 1 | **Hash-only filtering**<br>The 21041 interprets the descriptor buffer as a setup 512-bit hash table. If an incoming receive packet destination address is multicast or physical, the 21041 executes an imperfect address filtering against the hash table.<br>This field is valid only when setup packet (TDES1<27>) is set (Table 3–40). |

#### 4.2.2.3 Transmit Descriptor 2 (TDES2)

Figure 4–10 shows the TDES2 bit field, and Table 4–9 describes the bit field.

**Figure 4–10 TDES2 Transmit Descriptor 2**

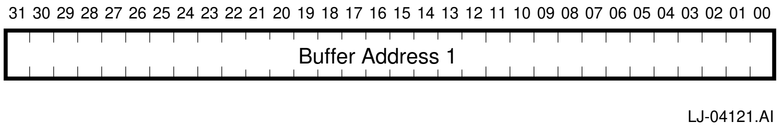



**Table 4–9 TDES2 Transmit Descriptor 2 Description**

| Field | Description |
|---|---|
| **31:0** | **Buffer Address 1**<br>Physical address of buffer 1. |

#### 4.2.2.4 Transmit Descriptor 3 (TDES3)

Figure 4–11 shows the TDES3 bit field, and Table 4–10 describes the bit field.

**Figure 4–11 TDES3 Transmit Descriptor 3**

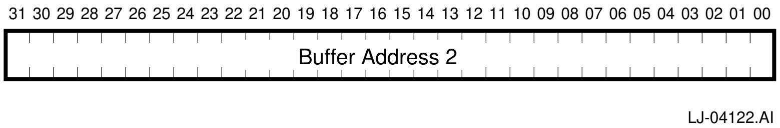



**Table 4–10 TDES3 Transmit Descriptor 3 Description**

| Field | Description |
|---|---|
| **31:0** | **Buffer Address 2**<br>Physical address of buffer 2. |

#### 4.2.2.5 Transmit Descriptor Status Validity

Table 4–11 lists the validity of the transmit descriptor status bits during transmission completion status. The following list defines the abbreviations used in the table.

LO—Loss of carrier (TDES0<11>)
NC—No carrier (TDES0<10>)
LC—Late collision (TDES0<9>)
EC—Excessive collisions (TDES0<8>)
HF—Heartbeat fail (TDES0<7>)
CC—Collision count (TDES0<6:3>)
ES—Error summary (TDES0<15>)
TO—Transmit jabber timeout (TDES0<14>)
UF—Underflow error (TDES0<1>)
DE—Deferred (TDES0<0>)

**Table 4–11 Transmit Descriptor Status Validity**

| | Transmit Status Report | | | | | | |
|---|---|---|---|---|---|---|---|
| **Transmission Status** | **LO** | **NC** | **LC** | **EC** | **HF** | **CC** | **(ES, TO, UF, DE)** |
| Underflow | NV | NV | V | V | NV | V | V |
| Excessive collisions | V | V | V | V | V | NV | V |
| Watchdog timeout | NV | V | NV | NV | NV | V | V |
| Internal loopback | NV | NV | V | V | NV | V | V |

V—Valid
NV—Not valid

### 4.2.3 Setup Frame

A setup frame defines the 21041 Ethernet destination addresses. These addresses filter all incoming frames. The setup frame is *never* transmitted on the Ethernet wire nor is it looped back to the receive list. When processing the setup frame, the receiver logic temporarily disengages from the Ethernet wire. The setup frame size must be *exactly* 192 bytes.

**Note**

The setup frame must be allocated in a single buffer that is longword-aligned. First segment (TDES1<29>) and last segment (TDES1<30>) must both be 0.

When the setup frame load is completed, the 21041 closes the setup frame descriptor by clearing its ownership bit and setting all other bits to 1.

#### 4.2.3.1 First Setup Frame

A setup frame must be processed before the reception process is started, except when it operates in promiscuous filtering mode.

#### 4.2.3.2 Subsequent Setup Frames

Subsequent setup frames may be queued to the 21041 regardless of the reception process state. To ensure correct setup frame processing, a packet may be queued at the beginning of the transmit descriptors ring or following a descriptor with zero length buffer. In the former case, the descriptor should carry the following information:

| | | |
|---|---|---|
| TDES0<31> | = | 1 - Adapter-owned descriptor |
| TDES1<10:0> | = | 0 - Transmit buffer 1 empty |
| TDES1<21:11> | = | 0 - Transmit buffer 2 empty |
| TDES1<29> | = | 0 - First segment bit 0 |
| TDES1<30> | = | 0 - Last segment bit 0 |

In the zero-length descriptor, TDES1<27> (setup packet) may also be set, in which case the address filtering bits TDES1<22> and TDES1<28> should be the same as in the previous setup packet.

The only requirement for setup frame processing is that the transmission process must be *running*. The setup frame is processed after all preceding frames have been transmitted and the current frame reception, if any, is completed.

The setup frame does not affect the reception process state, but during setup frame processing, the 21041 is disengaged from the Ethernet wire.

#### 4.2.3.3 Perfect Filtering Setup Frame Buffer

This section describes how the 21041 interprets a setup frame buffer in perfect filtering mode (CSR6<0> = 0).

The 21041 can store 16 destination addresses (full 48-bit Ethernet addresses). The 21041 compares the addresses of any incoming frame to these addresses and tests the status of the inverse filtering (CSR6<4>). It rejects addresses that

- Do not match if inverse filtering (CSR6<4> = 0) occurs
- Match if inverse filtering (CSR6<4> = 1) occurs

The setup frame must *always* supply all 16 addresses. Any mix of physical and multicast addresses can be used. Unused addresses should duplicate one of the valid addresses.

Figure 4–12 shows the perfect filtering setup frame buffer format of the addresses.

**Figure 4–12 Perfect Filtering Setup Frame Buffer Format**

| | 31 … 16 | 15 … 0 |
|---|---|---|
| <3:0> | xxxxxxxxxxxxxxxxxxxxxxxxxxx | Physical Address 00 (Bytes <1:0>) |
| <7:4> | xxxxxxxxxxxxxxxxxxxxxxxxxxx | Physical Address 00 (Bytes <3:2>) |
| <11:8> | xxxxxxxxxxxxxxxxxxxxxxxxxxx | Physical Address 00 (Bytes <5:4>) |
| | xxxxxxxxxxxxxxxxxxxxxxxxxxx | Physical Address 01 |
| | xxxxxxxxxxxxxxxxxxxxxxxxxxx | Physical Address 01 |
| | xxxxxxxxxxxxxxxxxxxxxxxxxxx | Physical Address 01 |
| | xxxxxxxxxxxxxxxxxxxxxxxxxxx | Physical Address 02 |
| | xxxxxxxxxxxxxxxxxxxxxxxxxxx | Physical Address 02 |
| | xxxxxxxxxxxxxxxxxxxxxxxxxxx | Physical Address 02 |
| | . . . | Physical Address 03 |
| | xxxxxxxxxxxxxxxxxxxxxxxxxxx | Physical Address 14 |
| | xxxxxxxxxxxxxxxxxxxxxxxxxxx | Physical Address 14 |
| | xxxxxxxxxxxxxxxxxxxxxxxxxxx | Physical Address 14 |
| <183:180> | xxxxxxxxxxxxxxxxxxxxxxxxxxx | Physical Address 15 (Bytes <1:0>) |
| <187:184> | xxxxxxxxxxxxxxxxxxxxxxxxxxx | Physical Address 15 (Bytes <3:2>) |
| <191:188> | xxxxxxxxxxxxxxxxxxxxxxxxxxx | Physical Address 15 (Bytes <5:4>) |

xxxxxx = don't care

MLO-010327

The low-order bit of the low-order bytes is the multicast bit of the address.

Example 4–1 shows a perfect filtering setup buffer (fragment).

**Example 4–1 Perfect Filtering Buffer**

```
Ethernet addresses to be filtered:
➊ A8-09-65-12-34-76
  09-BC-87-DE-03-15
    .
    .
    .

Setup frame buffer fragment while in little endian byte ordering:
➋ xxxx09A8
  xxxx1265
  xxxx7634
  xxxxBC09
  xxxxDE87
  xxxx1503
    .
    .
    .
Setup frame buffer fragment while in big endian byte ordering:
➌ A809xxxx
  6512xxxx
  3476xxxx
  09BCxxxx
  87DExxxx
  0315xxxx
    .
    .
    .
```

➊ This displays two Ethernet addresses written according to the Ethernet specification for address display.

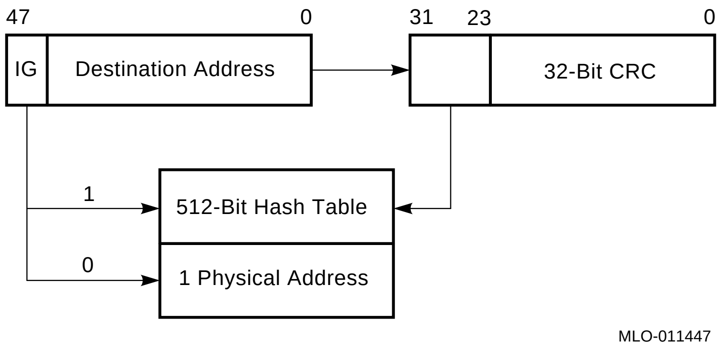
➋ This displays two addresses as they would appear in the buffer in little endian format.

➌ This displays two addresses as they would appear in the buffer in big endian format.

#### 4.2.3.4 Imperfect Filtering Setup Frame Buffer

This section describes how the 21041 interprets a setup frame buffer in imperfect filtering mode (CSR6<0> is set). Figure 4–13 shows imperfect filtering.

**Figure 4–13 Imperfect Filtering**



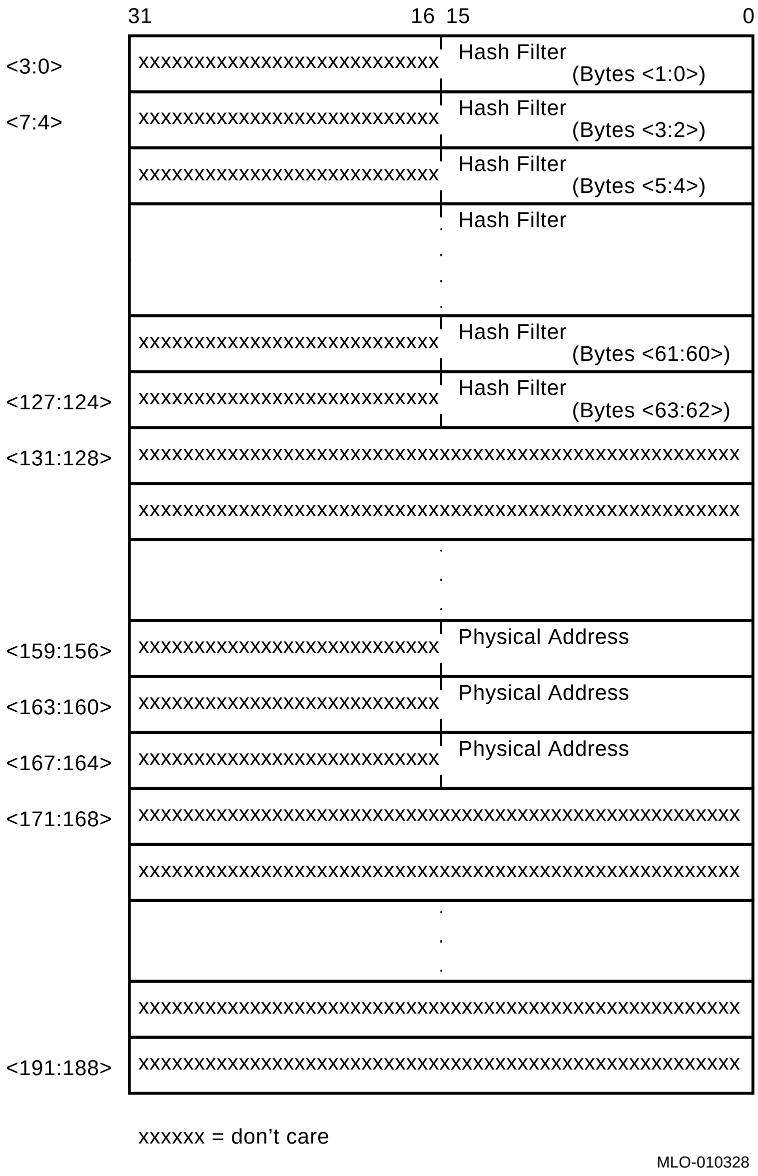

The 21041 can store 512 bits serving as a hash table and one *physical* 48-bit Ethernet address. Incoming frames with multicast destination addresses are subjected to imperfect filtering. Frames with physical destination addresses are checked against the single physical address.

For any incoming frame with a multicast destination address, the 21041 applies the standard Ethernet cyclic redundancy check (CRC) function to the first 6 bytes that contain the destination address, then it uses the most significant 9 bits of the result as a bit index into the table. If the indexed bit is set, the frame is accepted. If the bit is cleared, the frame is rejected. (Appendix C provides an example of a hash index for a given Ethernet address.)

This filtering mode is called imperfect because multicast frames not addressed to this station may slip through, but it still decreases the number of frames that the host can receive.

Figure 4–14 shows the format for the hash table and the physical address.

**Figure 4–14 Imperfect Filtering Setup Frame Format**




Bits are sequentially numbered from right to left and down the hash table. For example, if the CRC (destination address) <8:0> = 33, the 21041 examines bit 1 in the fourth longword.

Example 4–2 shows an imperfect filtering setup frame buffer.

**Example 4–2 Imperfect Filtering Buffer**

```
  Ethernet addresses to be filtered:
❶   25-00-25-00-27-00
    A3-C5-62-3F-25-87
    D9-C2-C0-99-0B-82
    7D-48-4D-FD-CC-0A
    E7-C1-96-36-89-DD
    61-CC-28-55-D3-C7
    6B-46-0A-55-2D-7E

❷   A8-12-34-35-76-08


  Setup frame buffer while in little endian byte ordering:

❸   xxxx0000
    xxxx0000
    xxxx0000
    xxxx1000
    xxxx0000
    xxxx0000
    xxxx0000
    xxxx0000
    xxxx0000
    xxxx0000
    xxxx0000
    xxxx4000
    xxxx0080
    xxxx0000
    xxxx0000
    xxxx0010
```

(continued on next page)

**Example 4–2 (Cont.) Imperfect Filtering Buffer**

```
   xxxx0000
   xxxx0000
   xxxx0000
   xxxx1000
   xxxx0000
   xxxx0000
   xxxx0000
   xxxx0000
   xxxx0000
   xxxx0000
   xxxx0000
   xxxx0001
   xxxx0000
   xxxx0000
   xxxx0000
   xxxx0040
   xxxxxxxx
   xxxxxxxx
   xxxxxxxx
   xxxxxxxx
   xxxxxxxx
   xxxxxxxx
   xxxxxxxx

❹  xxxx12A8
   xxxx3534
   xxxx0876
   xxxxxxxx
   xxxxxxxx
   xxxxxxxx
   xxxxxxxx
   xxxxxxxx
   xxxxxxxx
   xxxxxxxx
   xxxxxxxx
```

(continued on next page)

**Example 4–2 (Cont.) Imperfect Filtering Buffer**

```
Setup frame buffer while in big endian byte ordering:
❺ 0000xxxx
  0000xxxx
  0000xxxx
  0010xxxx
  0000xxxx
  0000xxxx
  0000xxxx
  0000xxxx
  0000xxxx
  0000xxxx
  0000xxxx
  0040xxxx
  8000xxxx
  0000xxxx
  0000xxxx
  1000xxxx
  0000xxxx
  0000xxxx
  0000xxxx
  0010xxxx
  0000xxxx
  0000xxxx
  0000xxxx
  0000xxxx
  0000xxxx
  0000xxxx
  0000xxxx
  0100xxxx
  0000xxxx
  0000xxxx
  0000xxxx
  4000xxxx
  xxxxxxxx
  xxxxxxxx
  xxxxxxxx
  xxxxxxxx
  xxxxxxxx
  xxxxxxxx
  xxxxxxxx
```

(continued on next page)

**Example 4–2 (Cont.) Imperfect Filtering Buffer**

```
❻ A812xxxx
   3435xxxx
   7608xxxx
   xxxxxxxx
   xxxxxxxx
   xxxxxxxx
   xxxxxxxx
   xxxxxxxx
   xxxxxxxx
   xxxxxxxx
   xxxxxxxx
```

❶ This displays Ethernet multicast addresses written according to the Ethernet specification for address display.

❷ This displays an Ethernet physical address.

❸ This displays the first part of an imperfect filter setup frame buffer, in little endian byte ordering, with set bits for the multicast addresses as in ❶.

❹ This displays the second part of the buffer with the physical address as in ❷, in little endian byte ordering.

❺ This displays the first part of an imperfect filter setup frame buffer, in big endian byte ordering, with set bits for the multicast addresses as in ❶.

❻ This displays the second part of the buffer with the physical address as in ❷, in big endian byte ordering.

# 5
# Functional Description

This chapter describes the reset commands, interrupt handling, and startup. It also explains the transmit and receive processes.

The functional operation of the 21041 is controlled by the driver interface located in the host communication area. The driver interface activity is controlled by command and status registers (CSRs), descriptor lists, and data buffers.

Descriptor lists and data buffers, collectively referred to as the host communication area, reside in host memory. These data structures process the actions and status related to buffer management. The 21041 transfers frame data to and from the receive and transmit buffers in host memory. Descriptors resident in the host memory act as pointers to these buffers (Chapter 4).

## 5.1 Reset Commands

Two commands are available to reset the 21041: hardware and software.

1. Assert **rst_l**, to initiate a hardware reset.
2. Assert **csr0 bit 0**, to initiate a software reset.

For a proper reset operation, both clocks (**clk_l**, **xtal1**) should operate normally. For both the hardware and software reset commands, the 21041 *aborts* all processing and starts the reset sequence. The 21041 initializes all internal states and registers.

**Note**

No internal states are retained, no descriptors are kept in the 21041, and all the host-visible registers are set to the reset values. However, a software reset command has no effect on the configuration registers.

The 21041 does not explicitly disown any owned descriptor; descriptor OWN bits can be left in a state indicating 21041 ownership. Sections 4.2.1.1 and 4.2.2.1 provide detailed descriptions of OWN bits.

---

After either a hardware or software reset command, the first bus transaction to the 21041 should not be initiated for at least 50 PCI clock cycles. When the reset sequence completes, the 21041 can accept host commands. The receive and transmit processes are placed in the stopped state (Table 5–2 and Table 5–3). It is permissible to issue successive reset commands (hardware or software).

## 5.2 Arbitration Scheme

In general, the arbitration scheme used by the 21041 grants precedence to the receive process instead of the transmit process (CSR0<1>). Table 5–1 lists a description of the arbitration scheme. The following list describes the technical expressions that appear in the table:

- **Txreq**—A DMA request for the transmit process when any one of the following conditions is satisfied:
  - Fetch descriptor
  - Close descriptor
  - Setup packet
  - Sufficient space allocated in the transmit FIFO for a full data burst
- **Rxreq**—A DMA request for the receive process when any one of the following conditions is satisfied:
  - Fetch descriptor
  - Close descriptor
  - Sufficient data entered into the receive FIFO for a full data burst, **or** the end of receive packet (Rxpacket) and the data in the receive FIFO is less than a full burst
- **Txen**—21041 is currently transmitting.

- **RxF>tr**— The amount of free bytes left in the receive FIFO. The values are taken from the programmed threshold values in CSR6<15:14>. Table 3–39 lists the coding for the programmed values which include 72, 96, 128, and 160 bytes.
- **TxF<tr**—The data in the transmit FIFO is less than the programmed transmit FIFO threshold. Table 3–39 lists the coding for the programmed values which include 72, 96, 128, and 160 bytes.

**Table 5–1 Arbitration Scheme**

| Txreq | Rxreq | Txen | RxF>tr | TxF<tr | Chosen Process |
|---|---|---|---|---|---|
| 0 | 0 | 0 | – | – | – |
| 0 | 0 | 1 | – | – | – |
| 0 | 1 | 0 | – | – | Receive process |
| 0 | 1 | 1 | – | – | Receive process |
| 1 | 0 | 0 | – | – | Transmit process |
| 1 | 0 | 1 | – | – | Transmit process |
| 1 | 1 | 0 | – | – | Receive process |
| 1 | 1 | 1 | 0 | 0 | Transmit process |
| 1 | 1 | 1 | 0 | 1 | Transmit process |
| 1 | 1 | – | 1 | 0 | Receive process |
| 1 | 1 | 1 | 1 | 1 | Receive process. If full-duplex is active, round-robin arbitration scheme is used. |

In addition to the arbitration scheme listed in Table 5–1, two other factors must be considered:

- The transmit process obtains a window for one burst between two consecutive receive packets.
- The receive process obtains a window for one burst between two consecutive transmit packets.

## 5.3 Interrupts

Interrupts can be generated as a result of various events. CSR5 contains all the status bits that may cause an interrupt. The following list contains the events that cause interrupts.

CSR5<0>—Transmit interrupt
CSR5<1>—Transmit process stopped
CSR5<2>—Transmit buffer unavailable
CSR5<3>—Transmit jabber timeout
CSR5<4>—Link pass
CSR5<5>—Transmit underflow
CSR5<6>—Receive interrupt
CSR5<7>—Receive buffer unavailable
CSR5<8>—Receive process stopped
CSR5<9>—Receive watchdog timeout
CSR5<11>—General-purpose timer expired
CSR5<12>—Link fail
CSR5<13>—System error
CSR5<14>—Early receive interrupt

Interrupt bits are cleared by writing a 1 to the bit position. This enables additional interrupts from the same source.

Interrupts are not queued, and if the interrupting event recurs *before* the driver has responded to it, no additional interrupts are generated. For example, receive interrupt (CSR5<6>) indicates that one or more received frames were delivered to host memory. The driver must scan *all* descriptors, from the last recorded position to the first one owned by the 21041.

An interrupt is generated only *once* for simultaneous, multiple interrupting events. The driver must scan CSR5 for the interrupt cause or causes. The interrupt is not generated *again*, unless a new interrupting event occurs after the driver has cleared the appropriate CSR5 bits.

For example, transmit interrupt (CSR5<0>) and receive interrupt (CSR5<6>) are set simultaneously. The host acknowledges the interrupt, and the driver begins executing by reading CSR5. Next, receive buffer unavailable (CSR5<7>) is set. The driver writes back its copy of CSR5, clearing transmit interrupt and receive interrupt. The interrupt line is deasserted for one cycle and then asserted again with receive buffer unavailable.

## 5.4 Startup Procedure

The following sequence of checks and commands must be performed by the driver to prepare the 21041 for operation.

1. Wait 50 PCI clock cycles for the 21041 to complete its reset sequence.
2. Update configuration registers (Section 3.1).
   - Read the configuration register to identify the 21041 and its revision.
   - Write the configuration interrupt register (if interrupt mapping is necessary).
   - Write the configuration command register.
   - Write the configuration register to map the 21041 I/O or memory address space into the appropriate processor address space.
   - Write the configuration latency counter to match the system latency guidelines.
3. Optionally, read the data stored in the serial ROM.
4. Write CSR0 to set global host bus operating parameters (Section 3.3.2).
5. Write the Interrupt Mask Register to mask unnecessary (depending on the particular application) interrupt causes.
6. Change the default settings of the jabber timers and SIA initial setting by writing to CSR13, CSR14, and CSR15 (Section 3.3.13).
7. The driver must create the transmit and receive descriptor lists. Then, it writes to both CSR3 and CSR4 providing the 21041 with the starting address of each list (Section 3.3.5). The first descriptor on the transmit list may contain a setup frame (Section 4.2.3).

**Caution**

If address filtering (either perfect or imperfect) is desired, the receive process should be started only after the setup frame has been processed (Section 4.2.3).

8. Write CSR6 (Section 3.3.7) to set global serial parameters and start both the receive and transmit processes. The receive and transmit processes enter the running state and attempt to acquire descriptors from the respective descriptor lists. Then the receive and transmit processes begin processing incoming and outgoing frames. The receive and transmit processes are independent of each other and can be started and stopped separately.

## 5.5 Receive Process

While in the running state, the receive process polls the receive descriptor list, attempting to acquire free descriptors. Incoming frames are processed and placed in acquired descriptors' data buffers. Status information is written to receive descriptor 0.

### 5.5.1 Descriptor Acquisition

The 21041 always attempts to acquire an extra descriptor in anticipation of incoming frames. Descriptor acquisition is attempted if any of the following conditions are satisfied:

- When start/stop receive (CSR6<1>) sets immediately after being placed in the running state.
- When the 21041 begins writing frame data to a data buffer pointed to by the current descriptor, and the buffer ends before the frame ends.
- When the 21041 completes the reception of a frame, and the current receive descriptor has been closed.
- When the receive process is suspended because of host-owned buffer (RDES0<31>=0), and a new frame is received.
- When receive poll demand is issued (Section 3.3.4).

### 5.5.2 Frame Processing

As incoming frames arrive, the 21041 recovers the incoming Manchester data, converts it to data and clock pulses, and then sends it to the receive engine. The receive engine strips the preamble bits and stores the frame data in the receive FIFO. Concurrently, the receive section performs address filtering depending on the results of inverse filtering (CSR6<6>), hash/perfect receive filtering mode (CSR6<0>), and hash-only receive filtering mode (CSR6<2>), as well as its internal filtering table. If the frame fails the address filtering, it is ignored and purged from the FIFO. Frames that are shorter than 64 bytes, because of collision or premature termination, are also ignored and purged from the FIFO (unless pass bad frames bit CSR6<3> is set).

After 64 bytes have been received, the 21041 requests the PCI bus in order to transfer the frame data to the buffer. The 21041 sets first descriptor (RDES0<9>) to delimit the frame. Then, the descriptors are released when the OWN (RDES0<31>) bit is reset to 0 either as the data buffers fill up or as the last segment of a frame is transferred to a buffer. If a frame is contained in a single descriptor, both last descriptor (RDES0<8>) and first descriptor (RDES0<9>) are set.

The 21041 sets last descriptor (RDES0<8>) and fetches the next descriptor, and the RDES0 status bits in the last frame descriptor are released. After the last frame descriptor is released, the 21041 sets receive interrupt (CSR5<6>) and fetches the next descriptor. The same process repeats unless the 21041 encounters a descriptor flagged as being owned by the host. If this occurs, the receive process sets receive buffer unavailable (CSR5<7>) and then enters the suspended state. The position in the receive list is retained.

### 5.5.3 Receive Process Suspended

If any frames enter while the receive process is suspended, the 21041 fetches the current descriptor in host memory. If the descriptor is now owned by the 21041, the receive process reenters the running state and starts the frame reception.

If a receive frame arrives while the receive process is suspended, the 21041 refetches the next descriptor. If the descriptor is still owned by the host, the 21041 increments the missed frames counter (CSR8<15:0>) and discards the current frame in the receive FIFO. If more than one frame is stored in the receive FIFO, the process repeats.

### 5.5.4 Receive Process State Transitions

Table 5–2 lists the receive process state transitions and the resulting actions.

**Table 5–2 Receive Process State Transitions**

| From State | Event | To State | Action |
|---|---|---|---|
| Stopped | Start receive command. | Running | Receive polling begins from last list position or from the list head, if this is the first start receive command issued or if the receive descriptor list address (CSR3) was modified by the driver. |

(continued on next page)

**Table 5–2 (Cont.) Receive Process State Transitions**

| From State | Event | To State | Action |
|---|---|---|---|
| Running | 21041 attempts to acquire a descriptor owned by the host. | Suspended | Receive buffer unavailable (CSR5<7>) sets when the last acquired descriptor buffer is consumed. The position in the list is retained. |
| Running | Stop receive command. | Stopped | Receive process is stopped after the current frame, if any, is completely transferred to data buffers. Receive process stopped (CSR5<8>) sets. The position in the list is retained. |
| Running | Memory or host bus parity error encountered. | Running | 21041 operation is stopped and system error (CSR5<13>) sets. The 21041 remains in the running state. A software reset must be issued to release the 21041. |
| Running | Reset command. | Stopped | Receive capability is cut off. |
| Suspended | Receive poll demand or incoming frame and available descriptor. | Running | Receive polling resumes from last list position. |
| Suspended | Stop receive command. | Stopped | Receive process stopped (CSR5<8>) sets. |
| Suspended | Reset command. | Stopped | None. |

## 5.6 Transmit Process

While in the running state, the transmit process polls the transmit descriptor list for frames requiring transmission. After polling starts, it continues in either sequential descriptor ring order or chained order. When it completes frame transmission, status information is written into transmit descriptor 0 (TDES0). If the 21041 detects a descriptor flagged as owned by the host, or if an error condition occurs, the transmit process is suspended and both transmit buffer unavailable (CSR5<2>) and normal interrupt summary (CSR5<16>) are set.

Transmit interrupt (CSR5<0>) is set after completing transmission of a frame that has interrupt on completion (TDES1<31>) set in its last descriptor. When this occurs, the transmission process continues to run.

While in the running state, the transmit process can simultaneously acquire two frames. As the transmit process completes copying the first frame, it immediately polls the transmit descriptor list for the second frame. If the second frame is valid, the transmit process copies the frame before writing the status information of the first frame.

### 5.6.1 Frame Processing

Frames can be data-chained and span several buffers. Frames must be delimited by the first descriptor (TDES1<29>) and the last descriptor (TDES1<30>) respectively.

As the transmit process starts execution, the first descriptor must have TDES1<29> set. When this occurs, frame data transfers from the host buffer to the internal FIFO. Concurrently, the transmit process attempts to acquire the next descriptor. The transmit process expects this descriptor to have TDES1<29> clear. If TDES1<30> is clear, it indicates an intermediary buffer. If TDES1<30> is set, it indicates the last buffer of the frame.

After the last buffer of the frame has been transmitted, the 21041 writes back the final status information to the transmit descriptor 0 (TDES0) word of the descriptor that has the last segment set in transmit descriptor 1 (TDES1<30>). At this time, if interrupt on completion (TDES1<31>) was set, the transmit interrupt (CSR5<0>) is set, the next descriptor is fetched, and the process repeats.

Actual frame transmission begins after the internal FIFO had reached either a programmable threshold CSR6<15:14> (Table 3–39) or after a full frame is contained in the FIFO. Descriptors are released (OWN bit TDES0<31> clears) when the 21041 completes the packet transmission.

### 5.6.2 Transmit Polling Suspended

Transmit polling can be suspended by either of the following conditions:

- The 21041 detects a descriptor owned by the host (TDES0<31>=0). To resume, the driver must give descriptor ownership to the 21041 and then, issue a poll demand command.
- A frame transmission is aborted when a locally induced error is detected. The appropriate transmit descriptor 0 (TDES0) bit is set.

If either of the previous two conditions occur, both abnormal interrupt summary (CSR5<15>) and transmit interrupt (CSR5<0>) are set, and the information is written to transmit descriptor 0 causing the suspension.

In both of the cases just described, the position in the transmit list is retained. The retained position is that of the *descriptor following the last descriptor closed* (set to host ownership) by the 21041.

**Note**

The 21041 does not automatically poll the transmit descriptor list. The driver must explicitly issue a transmit poll demand command after rectifying the suspension cause, unless the transmit automatic polling (CSR0<19:17>) field is not 0.

### 5.6.3 Transmit Process State Transitions

Table 5–3 lists the transmit process state transitions and the resulting actions.

**Table 5–3 Transmit Process State Transitions**

| From State | Event | To State | Action |
|---|---|---|---|
| Stopped | Start transmit command. | Running | Transmit polling begins from one of the following positions:<br>The last list position.<br>The head of the list, if this is the first start command issued after CSR4 was initialized or modified. |
| Running | 21041 attempts acquisition of a descriptor owned by the host. | Suspended | Transmit buffer unavailable (CSR5<2>) sets. |
| Running | Frame transmission aborts because a locally induced underflow error (TDES0<1>) is detected (Section 4.2.2.1). | Suspended | The following bits set:<br>TDES0<1>—Underflow error<br>CSR5<5> —Transmit underflow<br>CSR5<15>—Abnormal interrupt summary |

(continued on next page)

**Table 5–3 (Cont.) Transmit Process State Transitions**

| From State | Event | To State | Action |
|---|---|---|---|
| Running | Stop transmit command. | Stopped | Transmit process is stopped after the current frame, if any, is transmitted. |
| Running | Frame transmission aborts because a transmit jabber time-out (TDES0<14>) was detected (Section 4.2.2.1). | Stopped | The following bits are set:<br>TDES0<14>—Transmit jabber time-out<br>CSR5<1>—Transmit process stopped<br>CSR5<3>—Transmit jabber time-out<br>CSR5<15>—Abnormal interrupt summary |
| Running | Parity error detected by memory or host bus. | Running | Transmission is cut off and system error (CSR5<13>) sets. The 21041 remains in the running state. If a software reset occurs, normal operation continues. |
| Running | Reset command. | Stopped | Transmission is cut off. If CSR4 was not changed, the position in the list is retained. If CSR4 was changed, the next descriptor address is fetched from the header list (CSR4) when the poll demand command is issued. Transmit process stopped (CSR5<1>) sets. |
| Suspended | Transmit poll demand command issued. | Running | Transmit polling resumes from the last list position. |
| Suspended | Stop transmit command. | Stopped | Transmit process stopped (CSR5<1>) sets. |
| Suspended | Reset command. | Stopped | None. |

## 5.7 Loopback Operations

The 21041 supports two loopback modes: internal loopback and external loopback. The loopback data rate is 10 megabits per second.

### 5.7.1 Internal Loopback Mode

Internal loopback mode is normally used to verify that the internal logic operations function correctly. In loopback mode, the 21041 takes frames from the transmit list and loops them back internally to the receive list. In loopback mode, the 21041 disengages from the Ethernet wire. Internal loopback mode also supports the following modes of operation:

- Media access control (MAC) internal loopback mode in which transmit packets are looped back at the MAC level and the 21041 disengages the SIA. The loopback data rate is 10 megabits per second.
- 10BASE-T internal loopback mode in which transmit packets from the encoder output are selected and looped back to the decoder input. The loopback data rate is 10 megabits per second.

### 5.7.2 External Loopback Mode

External loopback mode is normally used to verify that the logic operations up to the Ethernet cable function correctly. In external loopback mode, the 21041 takes frames from the transmit list and transmits them on the Ethernet wire. Concurrently, the 21041 listens to the line that carries its own transmissions and places incoming frames in the receive list.

**Caution**

In external loopback mode, when transmitted frames are placed on the Ethernet wire, the 21041 does not check the origin of any incoming frames. It is possible for frames not originating from the 21041 to enter the receive buffers.

External loopback mode also supports the following modes of operation:

- 10BASE-T external loopback mode transmits packets using twisted-pair wires. Concurrently, the 21041 disables the internal collision detector and thus listens to the line that carries its own transmission.

  The board designer must use an external shunt to connect the transmit line with the receive line.
- AUI external loopback mode transmits packets using the AUI cable up-to-MAU (medium attachment unit) to check the MAU integrity.

### 5.7.3 Driver Entering Loopback Mode

To enter a specific loopback mode, the driver must take the following actions:

**Note**

All address filtering and validity checking rules apply while in either MAC or 10BASE-T mode.

---

1. Stop the receive and transmit processes by writing 0 to both the start/stop receive (CSR6<1>) and the start/stop transmit (CSR6<13>) fields. The driver must wait for any previously scheduled frame activity to cease by polling the transmit process state (<22:20>) and the receive process state (<19:17>) fields in CSR5.
2. Prepare the appropriate transmit and receive descriptor lists in host memory. These lists can follow the existing lists at the point of suspension or be new lists identified to the 21041 by the receive list base address in CSR3 and by the transmit list address in CSR4.
3. Stop the SIA by setting CSR13 to a value of 00000000H.
4. Program CSR13, CSR14, and CSR15 to the desired SIA operation mode according to Table 3–62.
5. Wait at least 5 microseconds.
6. Select the desired loopback mode according to Table 3–62.
7. Use start commands to place both the transmit and receive processes into the running state.
8. As in normal processing, execute any 21041 interrupts.

### 5.7.4 Driver Restoring Normal Operation

To restore normal operation, the driver must execute the following procedure:

1. Stop both the receive and transmit processes. The driver must wait for any previously scheduled frame activity to cease by polling both the transmit (CSR5<22:20>) and receive process state (CSR5<19:17>) fields in CSR5.
2. Prepare appropriate transmit and receive descriptor lists in host memory. These lists can either follow the existing lists at the point of suspension or be new lists that have to be identified to the 21041 by the receive list base address in CSR3 and the transmit list base address in CSR4.
3. Stop the SIA by setting CSR13 to a value of 00000000H.

4. Program CSR13, CSR14, and CSR15 to the desired SIA operation mode according to Table 3–62.
5. Wait at least 5 microseconds.
6. Select normal mode operation according to Table 3–62.
7. Use start commands to place both the transmit and receive processes into the running state.
8. Resume normal processing. Execute any 21041 interrupts.

## 5.8 Full-Duplex Operation

The 21041 supports full-duplex operation only in 10BASE-T mode. The 21041 activates the transmit and receive processes simultaneously. It also supports receive back-to-back packets with an interpacket gap (IPG) of 9.6 microseconds in parallel with transmit back-to-back packets with an IPG of 9.6 microseconds.

The 21041 implements a programmable full-duplex operating mode (CSR6<9>) bit that commands the MAC to ignore both the carrier and the collision detect signal. When the autonegotiation algorithm is used (CSR14<7>), the 21041 operates in full-duplex mode only if the negotiation results allow it. For additional information about programming full-duplex operation with autonegotiation, refer to Section 7.1.9.

The driver must take the following actions to enter full-duplex operation:

1. Stop the receive and transmit processes by writing 0 to CSR6<1> and CSR6<13> fields respectively. The driver must wait for any previously scheduled frame activity to cease by polling the transmit process state (CSR5<22:20>) and the receive process state (CSR5<19:17>).
2. Reset the SIA by writing 0 to CSR13.
3. Prepare appropriate transmit and receive descriptor lists in host memory. These lists can use the existing lists at the point of suspension or create new lists that must be identified to the 21041 by referencing the receive list base address in CSR3 and the transmit list base address in CSR4.
4. Set full-duplex mode (CSR6<9>) and operating mode (CSR6<11:10>).
5. Using Table 3–62 as a guide, set CSR13 through CSR15.
6. Wait for the link pass interrupt.

7. Place the transmit and receive processes in the running state by using the start commands.

8. Resume normal processing. Execute any 21041 interrupts.

## 5.9 Power-Saving Modes

The 21041 incorporates two different power-saving modes: sleep mode and snooze mode. The following subsections describe these power-saving modes.

### 5.9.1 Sleep Power-Saving Mode

Sleep mode can be activated when the 21041 is not being used (for example, not connected to the network) and it is important to reduce its power dissipation. While in sleep mode, most of the circuitries are disabled. This includes the DMA machine, FIFOs, RxM, TxM, SIA, twisted-pair and AUI interfaces, and the general-purpose timer. The PCI section is not affected and access to the 21041 configuration registers remains possible. Access to the 21041 CSRs is not allowed.

To enter sleep mode, the driver must take the following actions:

1. Stop the receive and transmit processes by writing 0 to the CSR6<1> and CSR6<13> fields, respectively. The driver must wait for any previously scheduled frame activity to cease. This is done by polling the transmit process state (CSR5<22:20>) and the receive process state (CSR5<19:17>).

2. Reset the SIA by writing 0 to CSR13<0>.

3. Set the CFDA<31> bit.

To exit sleep mode, the driver must take the following actions:

1. Clear CFDA<31>.

2. Wait 10 milliseconds.

3. Start the SIA by writing 1 to CSR13<0>.

4. Wait at least 5 microseconds.

5. Start the receive and transmit processes by writing 1 to the CSR6<1> and CSR6<13> fields respectively.

### 5.9.2 Snooze Power-Saving Mode

Snooze mode is a dynamic power-saving mode. When the snooze mode bit (CFDA<30>) is set, the 21041 reduces its power dissipation unless one or more of the following conditions is true:

- PCI slave or master access is conducted
- Transmit process is in the running state
- Receive process is in the running state but not waiting for a packet
- Receive FIFO is not empty
- MAC receive engine is not idle
- Carrier is sensed
- Link pass or link fail interrupt occurred

When none of these conditions is true, the 21041 disables all its internal circuitries except for the PCI interface and the SIA circuitry (not including the Manchester decoder that uses the 100-MHz phases). The 21041 automatically and immediately reenables all its circuitries when at least one of the following occurs:

- PCI slave access is conducted
- Carrier is sensed
- Link pass or link fail interrupt occurred

This results in the 21041 dynamically getting into and out of low-power mode, and overall power dissipation is reduced.

To activate snooze mode, the driver should set CFDA<30>, the snooze mode bit. To stop snooze mode, the driver should clear CFDA<30>.

**Note**

Snooze mode should not be used with boot ROM or serial ROM accesses. Also, snooze mode affects the general-purpose timer and the automatic poll demand functions and should not be used in this mode.

## 5.10 LED Support

The 21041 supports seven network event LEDs. These LEDs are connected to the boot ROM address/data lines (br_ad<6:0>) and do not require additional drivers. Table 5–4 describes the LEDs and pin connection.

**Table 5–4 LED Connection**

| Signal | Pin Number | Description |
|---|---|---|
| br_ad<0> | 119 | LinkPass |
| br_ad<1> | 118 | Transmit Jabber/General-Purpose LED 1 |
| br_ad<2> | 117 | TP Polarity/General-Purpose LED 2 |
| br_ad<3> | 116 | Receive |
| br_ad<4> | 115 | Receive Address Match |
| br_ad<5> | 114 | Collision |
| br_ad<6> | 113 | Transmit |

The Transmit Jabber, Receive, Receive Address Match, Collision, and Transmit LEDs are selectively stretched in a range of 52 to 78 milliseconds. CSR15<11> controls the stretching function.

The Transmit Jabber and TP Polarity LEDs can also be used as two general-purpose LEDs. The mode selection and their values are set in CSR15<15,14> and CSR15<7,6> respectively.

# 6
# Host Bus Operation

This chapter describes the bus slave and bus master read and write cycles. It also explains the initiation of termination cycles by the bus master and bus slave. Master termination cycles can be initiated by the 21041 and the memory controller.

## 6.1 Overview

The peripheral component interconnect (PCI) is the physical interconnection used between highly integrated peripheral controller components and the host system. The 21041 uses the PCI bus to communicate with the host CPU and memory.

The 21041 is directly compatible with the *PCI Local Bus Specification, Revision 2.0*. The 21041 supports a subset of the PCI bus cycles (transactions). When communicating with the host, the 21041 operates as a bus slave; when communicating with the memory, it acts as a bus master.

All signals are sampled on the rising edge of the clock. Each signal has a setup and hold aperture with respect to the rising clock edge. Refer to the *DECchip 21041 PCI Ethernet LAN Controller Data Sheet* for detailed timing information. Table 2–2 lists the codes for the bus commands.

**Note**

The term **clock cycle**, as used in this chapter, refers to the PCI bus clock period specification.

## 6.2 Bus Slave Operation

All host accesses to CSRs and configuration registers in the 21041 are executed with the 21041 acting as the slave. The bus slave operations include the following:

- I/O read
- I/O write
- Configuration read
- Configuration write
- Memory read
- Memory write
- Memory read/write (includes memory write and invalidate, memory read line, and memory read multiple)

**Note**

- The 21041 does not support the following bus transactions:

  Interrupt acknowledge
  Special cycle
  Dual-address cycle

- If the 21041 is targeted for a burst I/O or memory operation, it responds with a disconnect on the second data transaction.

---

### 6.2.1 Slave Read Cycle (I/O or Memory Target)

Figure 6–1 shows a typical slave read cycle. The 21041 I/O read cycle is executed as follows:

1. The host initiates the slave read cycle by asserting the frame_l signal, driving the address on the ad lines and driving the bus command (slave read operation) on the c_be_l lines.
2. The 21041 samples the address and the bus command on the next clock edge.
3. The host deasserts frame_l signal and asserts irdy_l signal.
4. The 21041 asserts devsel_l and, at the next cycle, drives the data on the ad lines.

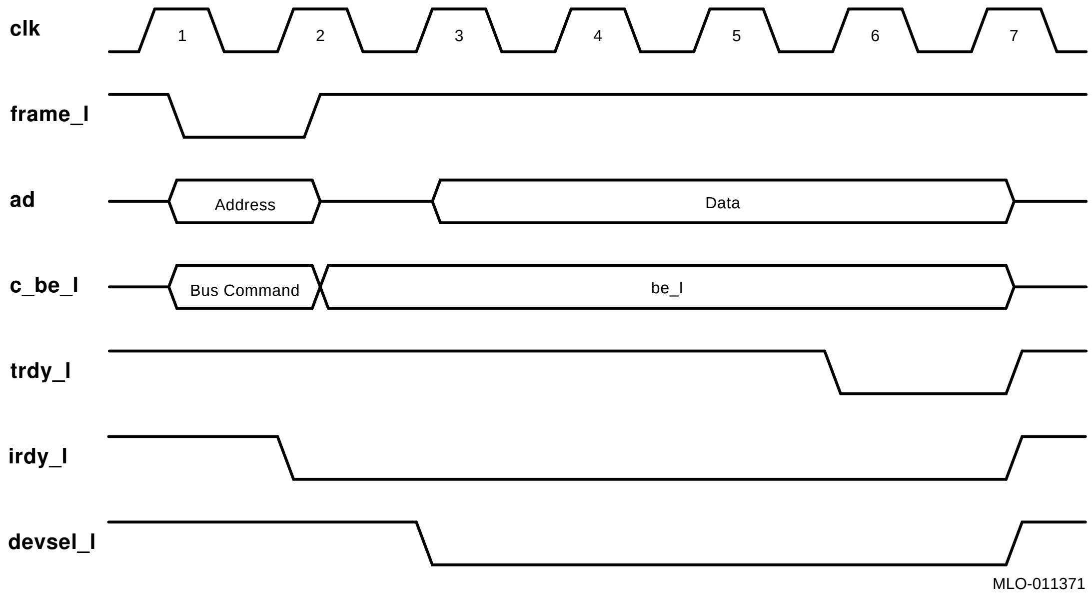
5. The read transaction completes when both irdy_l and trdy_l are asserted by the host and the 21041, respectively, on the same clock edge.

   The 21041 assumes that c_be_l lines are 0000 (longword access).

   If the c_be_l lines are 1111, the ad bus read is 00000000H with correct parity.

6. The host and the 21041 terminates the cycle by deasserting irdy_l and trdy_l respectively.

**Figure 6–1 Slave Read Cycle**




### 6.2.2 Slave Write Cycle (I/O or Memory Target)

Figure 6–2 shows a typical slave write cycle. The 21041 slave write cycle is executed as follows:

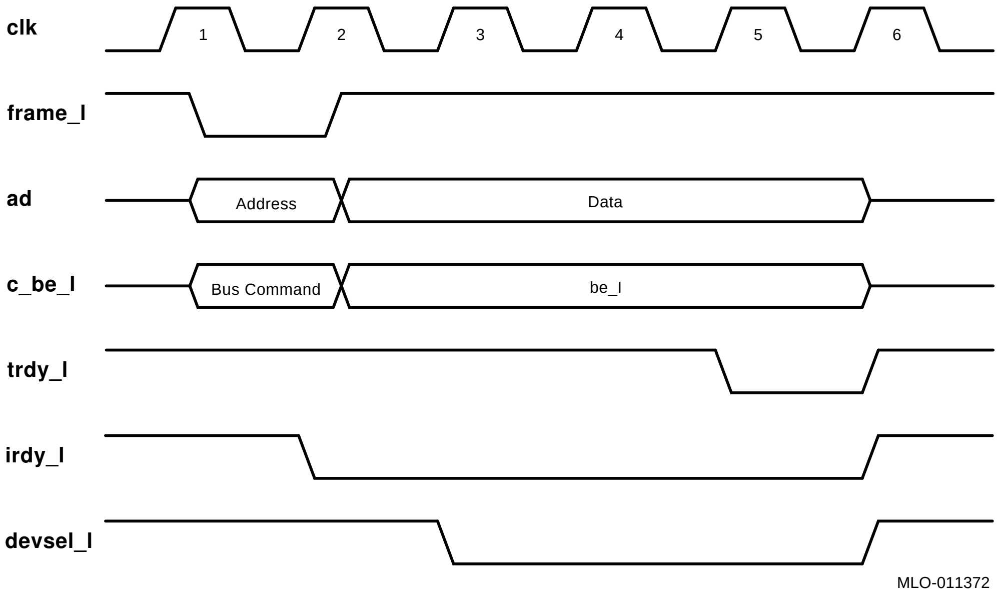
1. The host initiates the slave write cycle by asserting the frame_l signal and by driving the address on the ad lines and the bus command (slave write operation) on the c_be_l lines.
2. The 21041 samples the address and the bus command on the next clock edge.

3. The host deasserts frame_l and drives the data on the ad lines along with irdy_l.
4. The 21041 samples the data and asserts devsel_l and trdy_l.
5. The host and the 21041 complete the write transaction by asserting irdy_l and trdy_l, respectively, on the same clock edge.

   The 21041 assumes that c_be_l lines are 0000 (longword access).

   If the c_be_l lines are 1111, the write transaction completes normally on the bus, but the write to the CSR is not executed.
6. The host and the 21041 terminate the cycle by deasserting irdy_l and trdy_l respectively.

**Figure 6–2 Slave Write Cycle**



### 6.2.3 Configuration Read and Write Cycles

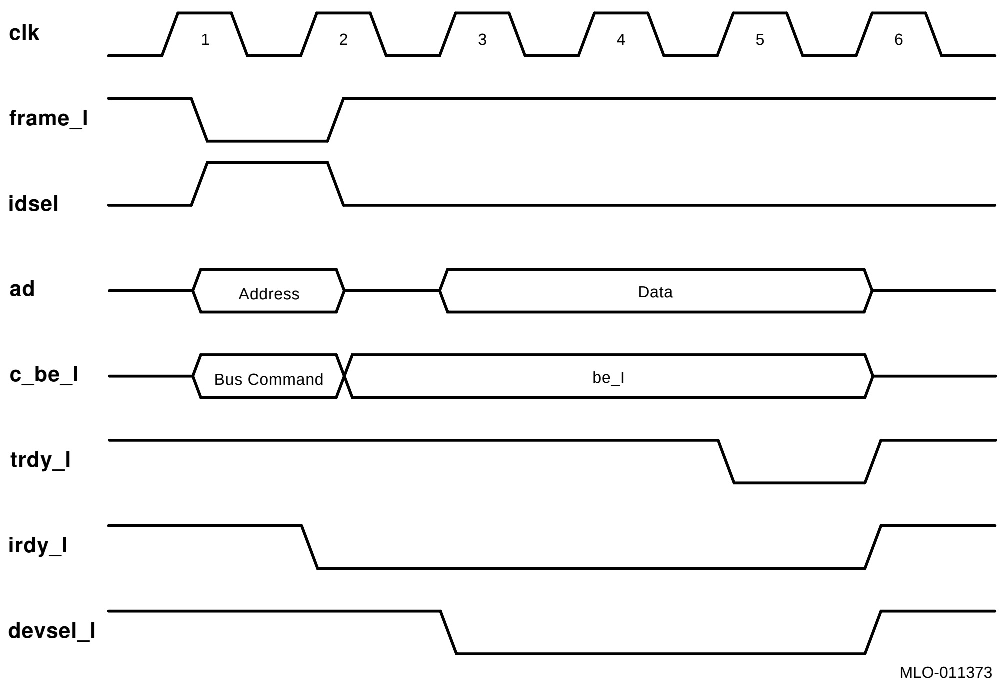
The 21041 provides a way for software to analyze and configure the system before defining any address assignments or mapping. The 21041 provides 256 bytes of configuration registers. Section 3.1 describes these registers.

**Note**

c_be_l lines are supported in the configuration space accesses.

Figure 6–3 shows a configuration read cycle. The hosts selects the 21041 by asserting idsel. The 21041 responds by asserting devsel_l. The remainder of the read cycle is similar to the slave read cycle (Section 6.2.1).

**Figure 6–3 Configuration Read Cycle**



## 6.3 Bus Master Operation

All memory accesses are completed with the 21041 as the master on the PCI bus. The bus master operations include the following:

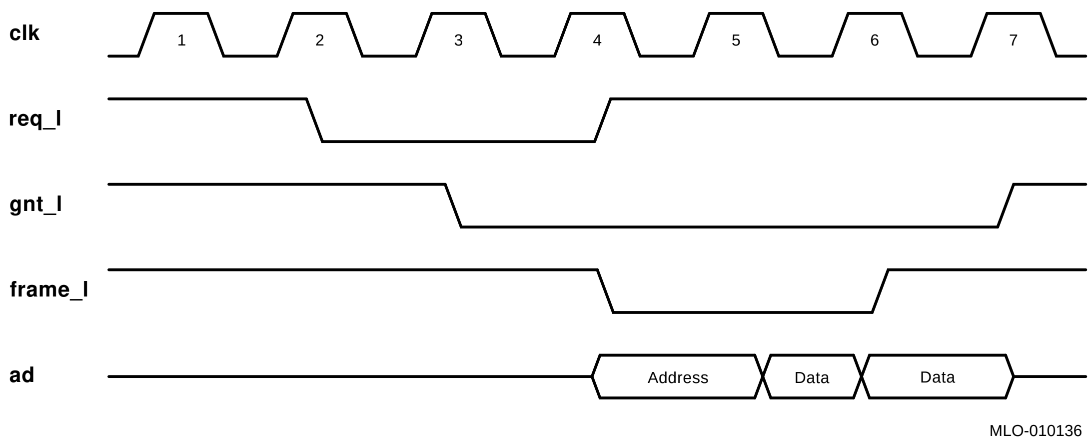
- Bus arbitration
- Memory read cycle
- Memory write cycle
- Terminations

### 6.3.1 Bus Arbitration

The 21041 uses the PCI central arbitration mechanism with its unique request (req_l) and grant (gnt_l) signals. Figure 6–4 shows the bus arbitration mechanism. The 21041 bus arbitration is executed as follows:

1. The 21041 requests the bus by asserting req_l.
2. The arbiter, in response, asserts gnt_l (gnt_l) can be deasserted on any clock).
3. The 21041 ensures that its gnt_l is asserted on the clock edge on which it wants to drive frame_l. (If gnt_l is deasserted, the 21041 does not proceed).
4. The 21041 deasserts req_l on the cycle that it asserts frame_l.

**Figure 6–4 Bus Arbitration**

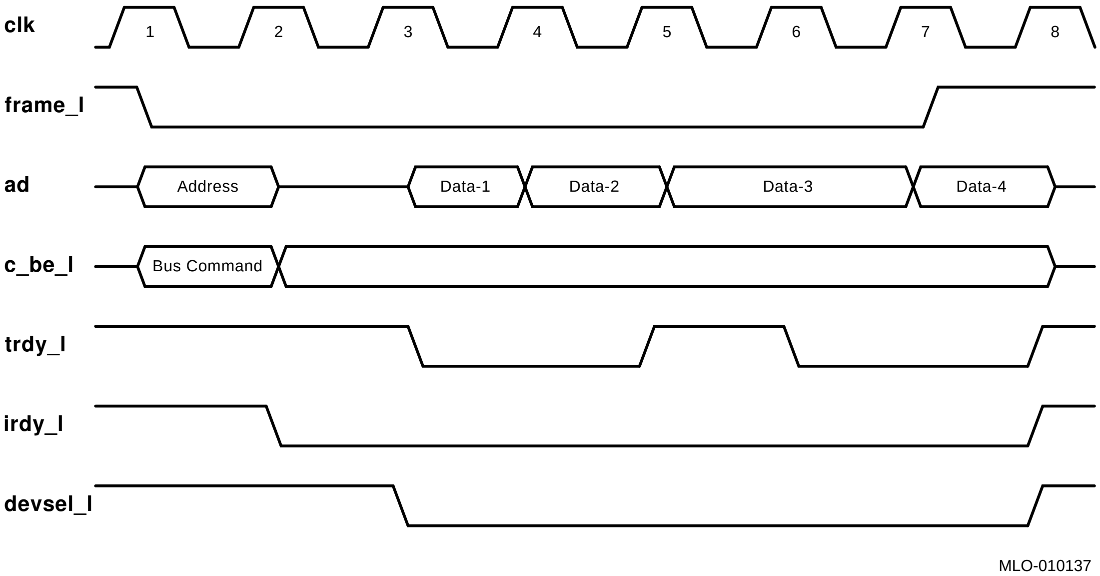



## 6.3.2 Memory Read Cycle

Figure 6–5 shows the memory read cycle. The memory read cycle is executed as follows:

1. The 21041 initiates the memory read cycle by asserting frame_l. It also drives both the address on the ad lines and the appropriate bus command (read operation) on the c_be_l lines.
2. The memory controller samples the address and the bus command on the next clock edge.
3. The 21041 asserts irdy_l until the end of the read transaction.
4. During the data transfer cycles, c_be_l indicates which byte lines are involved in each cycle. The 21041 drives 0000 on the c_be_l lines (longword access).
5. The memory controller drives the data on the ad lines and asserts trdy_l.
6. The 21041 samples the data on each rising clock edge when both irdy_l and trdy_l are asserted.
7. The previous two steps may be repeated a number of times.
8. The cycle is terminated when frame_l is deasserted by the 21041.
9. irdy_l is deasserted by the 21041 and trdy_l is deasserted by memory controller.

**Figure 6–5 Memory Read Cycle**



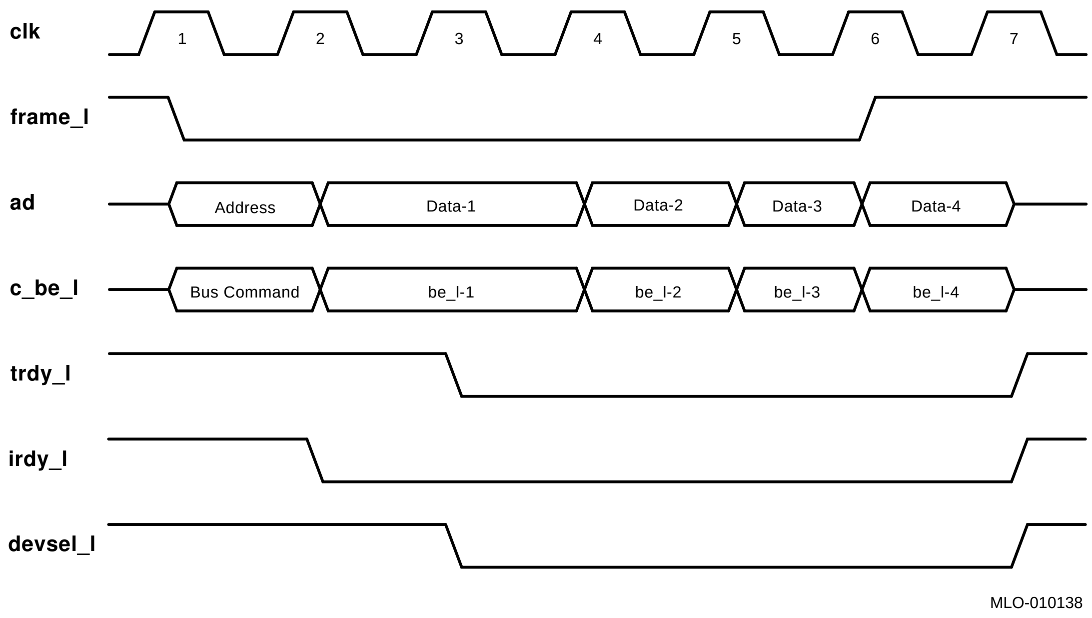

### 6.3.3 Memory Write Cycle

Figure 6–6 shows the memory write cycle. The memory write cycle is executed as follows:

1. The 21041 initiates the memory write cycle by asserting frame_l. It also drives both the address on the ad lines and the write operation bus command on the c_be_l lines.
2. The 21041 asserts irdy_l until the end of the transaction and drives the data on the ad lines.
3. The memory controller samples the address and the bus command on the next clock edge and asserts devsel_l.
4. During the data transfer, cycles c_be_l indicate which byte lanes are involved in each cycle. The 21041 drives 0000 on the c_be_l lines (longword access).
5. The memory controller samples the data and asserts trdy_l. Each data cycle is completed on the rising clock edge when both irdy_l and trdy_l are asserted.
6. The previous two steps may be repeated a number of times.

7. The 21041 terminates the cycle by deasserting frame_l.
8. The 21041 deasserts irdy_l and the memory controller deasserts trdy_l.

**Figure 6–6 Memory Write Cycle**



## 6.4 Termination Cycles

Termination cycles can be initiated during either slave or master cycles.

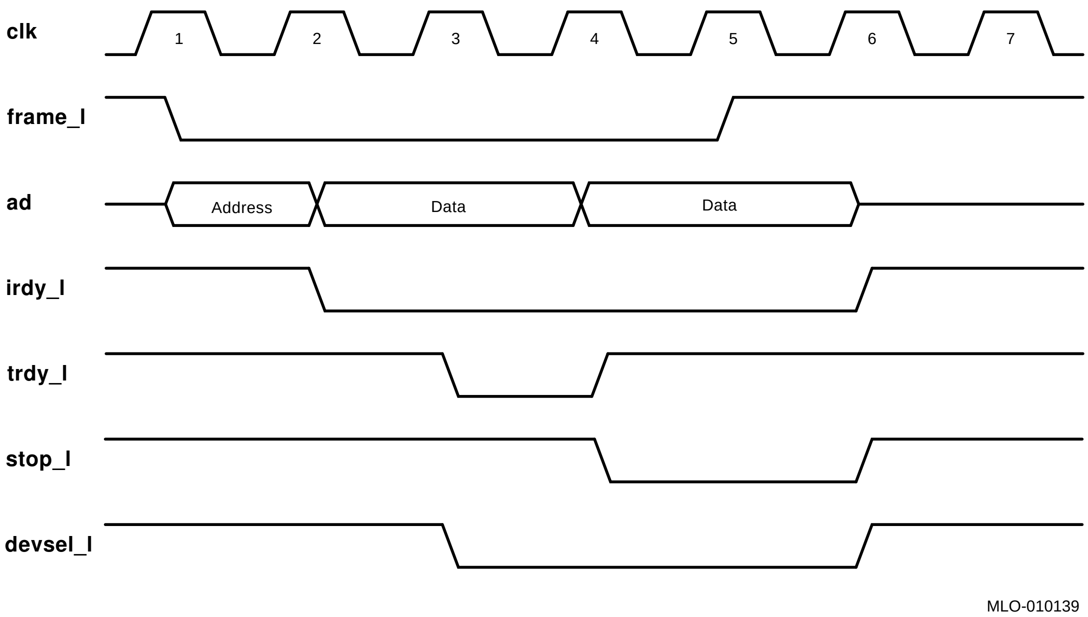
### 6.4.1 Slave-Initiated Termination

The 21041 initiates termination in slave mode when it is accessed by the host with I/O or memory burst cycles. The 21041 asserts stop_l to request that the host terminate the transaction. After stop_l is asserted, it remains asserted until frame_l is deasserted.

Figure 6–7 shows the retried device (the host) releasing the bus. The host retries the last data transaction after acquiring the bus in a different arbitration.

As a slave, the 21041 never initiates target abort.

**Figure 6–7 21041-Initiated Retry Cycle**



### 6.4.2 Master-Initiated Termination

A master-initiated termination can occur when the 21041 operates as a master device on the PCI bus. Terminations can be issued by either the 21041 or the memory controller.

21041 terminations include the following:

Normal completion
Time out
Master abort

Memory-controller terminations (target) include the following:

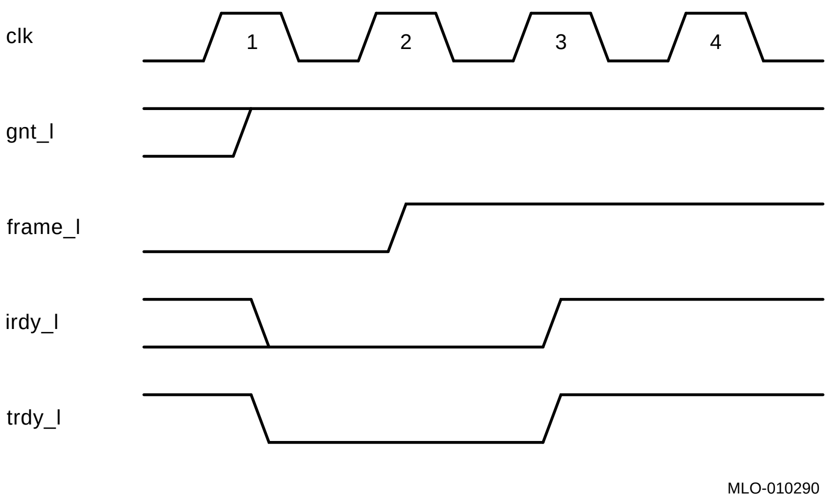
Target abort
Target-initiated termination
Target retry

#### 6.4.2.1 21041-Initiated Termination

A 21041-initiated termination occurs when frame_l is deasserted and irdy_l is asserted. This indicates to the memory controller that the final data phase is in progress. The final data transfer occurs when both irdy_l and trdy_l assert. The transaction completes when both frame_l and irdy_l deassert. This is an idle bus condition.

**6.4.2.1.1 Normal Completion** Figure 6–8 shows a normal completion cycle termination. This indicates that the 21041 successfully completed its intended transaction.

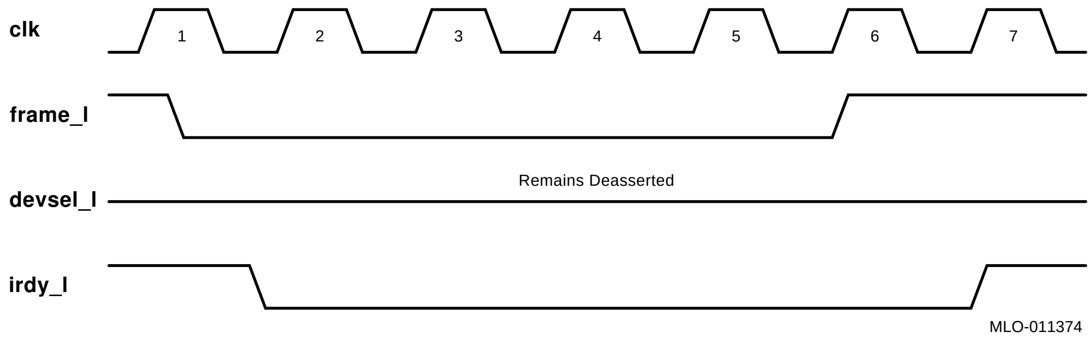
**Figure 6–8 Normal Completion Cycle**




**6.4.2.1.2 Time Out** A time-out cycle termination occurs when the gnt_l line has been deasserted by the arbiter and the 21041 internal latency timer has expired. However, the intended transaction has not completed. When timeout occurs, the 21041 performs normal transaction completion in which a maximum of two additional data phases are permitted.

**6.4.2.1.3 Master Abort** If the target does not assert devsel_l within five cycles from the assertion of frame_l, the 21041 aborts the cycle. It also asserts both master abort (CFCS<29>) and abnormal interrupt summary (CSR5<15>). Figure 6–9 shows the 21041 master abort cycle.

**Figure 6–9 Master Abort Cycle**

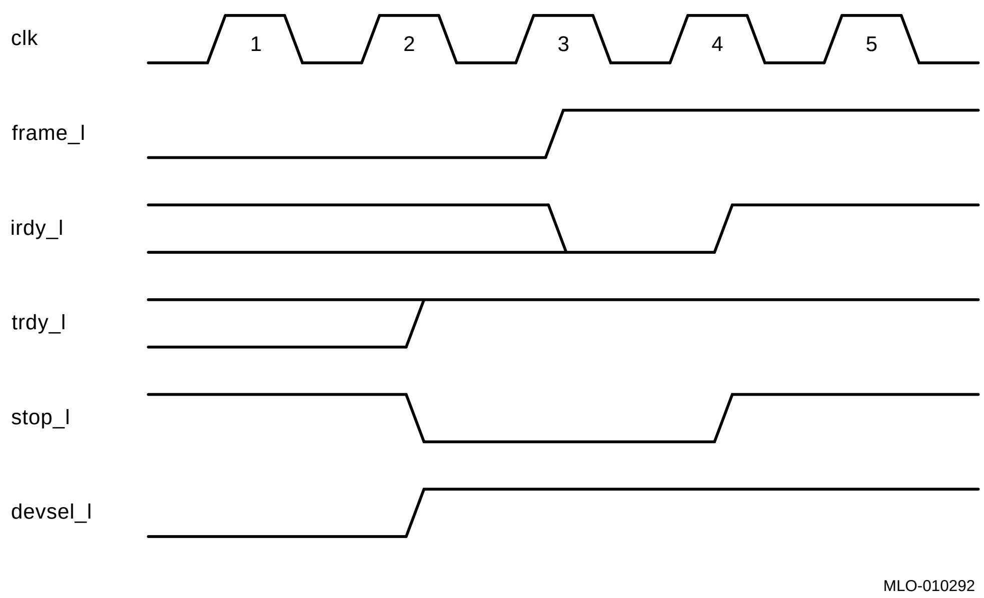



### 6.4.2.2 Memory-Controller-Initiated Termination

The memory controller or target can initiate certain terminations when the 21041 is the bus master.

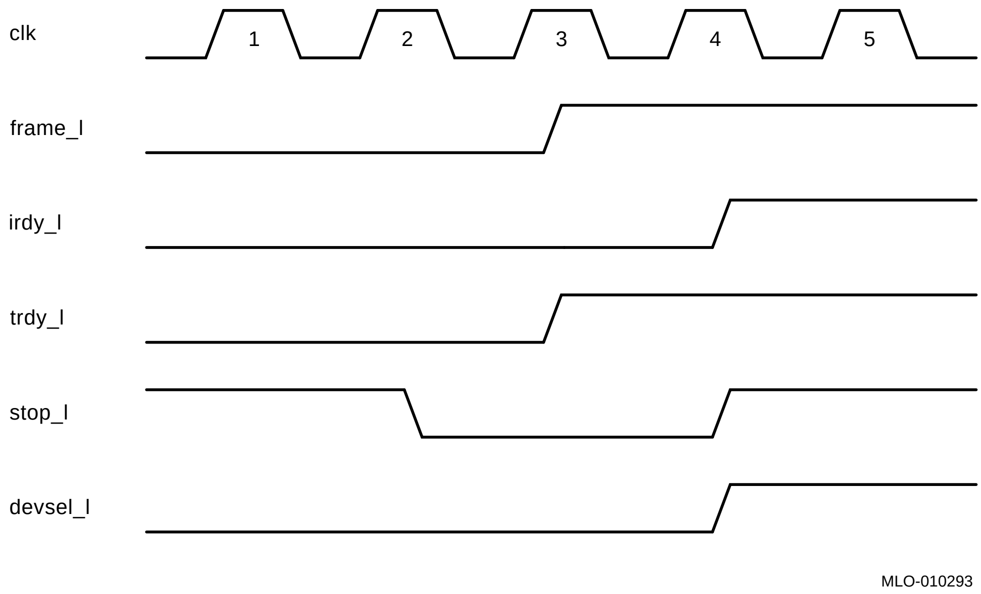
**6.4.2.2.1 Target Abort** The 21041 aborts the bus transaction when the target asserts stop_l and deasserts devsel_l. This indicates that the target wants the transaction to be aborted. The 21041 releases the bus and aborts the operation by asserting both received target abort (CFCS<28>) and abnormal interrupt summary (CSR5<15>). Figure 6–10 shows the 21041 abort cycle.

**Figure 6–10 Abort Cycle**



**6.4.2.2.2 Target-Initiated Termination** The 21041 terminates the bus transaction when the target asserts stop_l, which remains asserted until frame_l is deasserted. The 21041 releases the bus. Then, it retries at least the last data transaction after regaining the bus in another arbitration. Figure 6–11 shows the 21041 termination cycle.

**Figure 6–11 Termination Cycle**



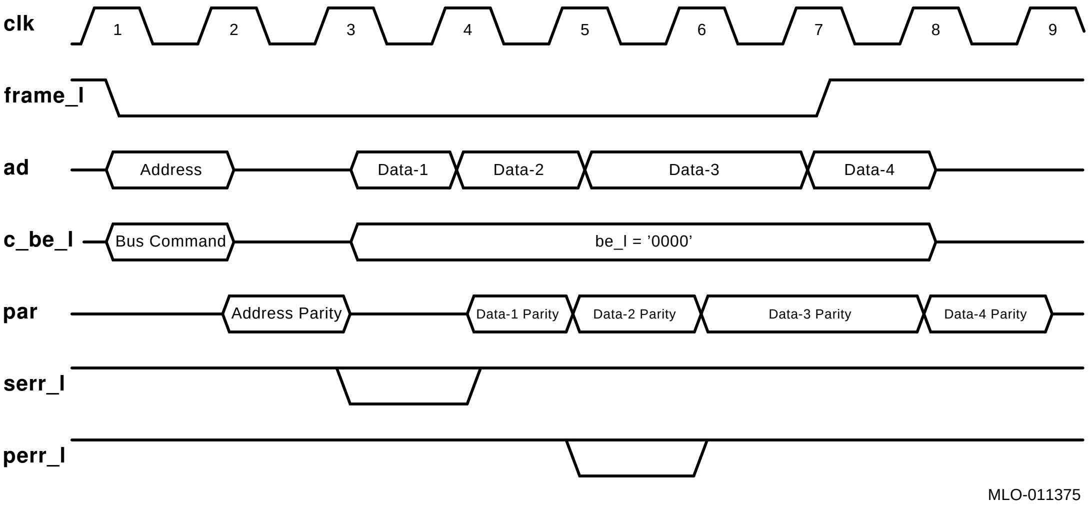

**6.4.2.2.3 Target Retry** The 21041 retries the bus transaction when the target asserts stop_l and deasserts trdy_l; stop_l remains asserted until frame_l is deasserted. The 21041 releases the bus. Then, it retries at least the last two data transactions after regaining the bus in another arbitration.

## 6.5 Parity

The 21041 supports parity generation on all address, data, and command bits. Parity is always checked and generated on the 32-bit address and data bus (ad) as well as on the four command (c_be_l) lines. The 21041 always transfers stable values (1 or 0) on all the ad and c_be_l lines. If a data parity is detected or perr_l is asserted when the 21041 is a bus master, the 21041 asserts detected parity error (CFCS<31>) and abnormal interrupt summary (CSR5<15>).

Figure 6–12 shows an example of parity generation on a memory write burst transaction. Note that valid parity is generated one cycle after the address and data segments were generated on the bus. One cycle after the assertion of the address parity, serr_l is asserted for one cycle because of an address parity error during slave operation. One cycle after the assertion of the data parity, perr_l is asserted because of a parity data error in either slave write or master read operations.

**Figure 6–12 Parity Operation Cycle**




## 6.6 Parking

Parking in the PCI bus allows the central arbiter to pause any selected agent. The 21041 enters the parking state when the arbiter asserts its gnt_l line while the bus is idle.

# 7
# Network Interface

This chapter describes the 10BASE-T or twisted-pair (TP) interface and the attachment unit interface (AUI).

**Note**

All frequency and timing information in this chapter is for 10 megabits per second serial operation.

## 7.1 10BASE-T and AUI Functions

The 10BASE-T and AUI functions include the following:

- Supports data driver and is receiver compatible with 10BASE-T specifications
- Supports data driver and is receiver compatible with AUI specifications
- Provides AUI collision receiver compatible with AUI specifications
- Selects either AUI or 10BASE-T interfaces
- Implements Manchester decoder for incoming data
- Implements Manchester encoder for outgoing data
- Contains onchip, 20-megahertz crystal oscillator circuitry
- Enables watchdog timers on incoming and outgoing data
- Contains 10BASE-T enhanced features that include:

- Smart squelch, rejecting noise detected by the 10BASE-T receiver interface
- Combined autopolarity and link test detection, presenting a robust algorithm for detection of both wire failure and switching of wires. Polarity correction is automatically done, while wire failure is reported to higher layers.

### 7.1.1 Receivers and Drivers

The host selects one set of data receivers and drivers at a time: either AUI or twisted-pair (TP). The other receiver and driver sets are disabled.

### 7.1.2 Manchester Decoder

The Manchester decoder is a phase-locked loop decoder that provides received clocks and data to the media access control (MAC) interface (Section 7.2).

### 7.1.3 Manchester Encoder

The Manchester encoder receives clocked data from the transmit engine and uses the 20-megahertz clock to provide Manchester encoded data. The encoder provides the transition to idle for the AUI and TP drivers.

### 7.1.4 Oscillator Circuitry

The 21041 supports two options for generating the internal 10-megahertz clock required by the internal circuitry.

1. An external parallel resonant crystal connected between **xtal1** and **xtal2** to drive the 21041-integrated oscillator circuitry.
2. An external clock generator module connected to **xtal1**; **xtal2** remains unconnected.

In both cases, the 21041 must be provided with a 20-megahertz signal that is internally divided by 2 to generate the 10-megahertz clock.

When driving the oscillator from an external clock source, an external clock having the following characteristics must be used to ensure proper operation of the 21041:

- Clock frequency: 20-megahertz $\pm$0.01% (TTL or CMOS)
- Rise/fall time: < 4 nanoseconds
- Duty cycle: 40%–60%

Table 7–1 lists the specifications for the crystal oscillator.

**Table 7–1 Crystal Oscillator Specification**

| Category | Value |
|---|---|
| Frequency | 20 megahertz |
| Tolerance | ±0.01% at 25°C |
| Stability | ±0.005% at 0°C to 70°C |

### 7.1.5 Jabber and Watchdog Timers

The jabber timer monitors the time of each packet transmission. The watchdog timer monitors the time of each packet reception. If a single packet transmission or reception exceeds a programmable value (Section 3.3.13.4), the jabber and watchdog circuitry automatically disables both the transmit and receive path. The transmit jabber timer provides the jabber function for 10BASE-T mode by cutting off transmission and asserting the collision signal to the MAC.

The packet descriptor closes with both transmit jabber time out (TDES0<14>) and late collision (TDES0<9>) setting if the jabber timer expires on a transmit packet.

### 7.1.6 Smart Squelch

The 21041 implements an intelligent squelch on its TP receiver to ensure that impulse noise detected on the receive inputs is not mistaken for valid signals. The squelch circuitry employs a combination of both amplitude and timing measurements to determine the validity of data received on the TP inputs.

The squelch circuit allows only valid differential receive data to pass through to the Manchester decoder provided that the following two conditions are satisfied:

1. The input amplitude is greater than the minimum signal threshold level.
2. A specific pulse sequence is received.

Satisfying these two conditions ensures that a good signal-to-noise ratio is maintained while the signal pair is active, and it prevents system noise from causing false squelch deactivation.

The line squelch quickly activates and deactivates within the specified time intervals, when the input squelch threshold is exceeded and a specific pulse sequence of proper polarity is detected.

The squelch circuitry rejects system noise by ignoring received pulses that are less than the required fixed time width. It also rejects pulses that are greater than the expected signal duration.

### 7.1.7 Autopolarity Detector

The autopolarity detector (CSR14<13>) provides a method of detecting wire polarity by switching the polarity of the data going into the MAC layer accordingly. To detect polarity, the 21041 uses the link test pulse and the end-of-frame delimiter in an algorithm integrated into the link integrity test, as specified in the IEEE 802.3 10BASE-T supplement.

### 7.1.8 Serial Port Autosensing

The 21041 can sense the AUI and TP ports at the same time. In addition, while the AUI port is used for transmission, it can also send the 10BASE-T link pulses onto the TP wires. These features, along with reported status bits and interrupts, allow the driver to choose between the AUI or TP port for network connection without any network configuration information.

To implement the auto-sensing algorithm, the driver uses the following hardware support provided by the 21041:

Interrupts

- Link pass CSR5<4>
- Link fail CSR5<12>
- Timer expired CSR5<11>

CSRs

- Autosensing enable bit CSR14<15>
- Activity sensed on the selected port CSR12<8>
- Activity sensed on the nonselected port CSR12<9>
- General-purpose timer (CSR11)

Selecting one of the serial ports requires programming of CSR13, CSR14, and CSR15. Table 3–62 provides the programming values for port autosensing enabled. To change the selection, start by resetting the SIA using CSR13.

When the driver selects the 10BASE-T port, it should set a time limit for the link pass interrupt to come. During this time, the BNC port is sensed and registered in CSR12<9>. If a link fail or timer expired interrupt then occurs, the driver should read that bit and change the port selection to AUI or BNC accordingly. If a link pass interrupt arrives, it means that 10BASE-T is the active port and the driver should stop the timer.

When the driver selects the AUI or BNC port, it should set a time limit for activity to be sensed on that port. When the timer expires, the driver should read CSR12<8> for indication of the selected port activity. If the bit is set, it means that the port is active. Otherwise, the driver selects the other port (AUI if the current selection is BNC and vice versa) and sets the timer again. When the AUI or BNC port is selected and a link pass interrupt arrives, the driver should switch the port selection to 10BASE-T.

The driver can use the general-purpose timer provided by the 21041. It can also use a dedicated timer or general cyclic timer if such timers are provided by the operating system.

### 7.1.9 Half- or Full-Duplex Autonegotiation

The IEEE 802.3 100BASE-T autonegotiation algorithm allows a device to advertise enhanced modes of operation it possesses to a device at the remote end of a link segment. Similarly, a device can detect corresponding enhanced operation modes that the other device may be advertising. The algorithm builds upon the existing 10BASE-T link pulse scheme and is based on data exchange in the physical layer between two nodes.

The 21041 implements this algorithm for half- and full-duplex mode autodetection while the 10BASE-T port is selected. The whole negotiation is done by the 21041 without software involvement. At the end of the negotiation, the 21041 notifies the host of the agreed operation mode (half- or full-duplex).

To enable the autonegotiation mechanism, CSR14<7> (autonegotiation enable) must be set. Table 3–62 shows the programming of the SIA with autonegotiation enabled.

Before enabling its receive or transmit paths, or after the Link Integrity Test has failed, the 21041 starts an autonegotiation sequence with its link partner. The 21041 stops sending its link pulses for at least 1.2 seconds and moves its link partner into the link fail state, forcing it to re-negotiate. The 21041 advertises its ability to operate in full-duplex only if CSR6<9> (full-duplex bit) is set. The full-duplex mode is adopted if both devices advertised support for it.

A link pass interrupt, together with CSR12<14:12> read as 101#2, indicates the end of the negotiation. The driver reads CSR12 to get the completion status and the driver also has the ability to restart the negotiation by setting CSR12<10>.

### 7.1.10 10BASE-T Link Integrity Test

Before transmitting on an Ethernet CSMA/CD network, each device has to check the reliability of its receive lines. For the AUI connection, this is indicated by the carrier signal present during transmission. In the twisted-pair (TP) case, link pulses are sent every 8 to 24 milliseconds at the interval between two transmissions.

The 21041 monitors the received link pulses and end-of-frame delimiters to be spaced and electrically shaped as specified in the IEEE 802.3 10BASE-T supplement. Accordingly, the 21041 implements the Link Integrity Test.

After a software or hardware reset, the 21041 wakes up in the link fail state. In this state, only link pulses are sent onto the transmit lines. Upon detection of the required line activity, the 21041 enters the link pass state which includes notifying the host by a link pass interrupt and enabling the receive and transmit paths.

A broken or noisy wire can bring the 21041 back to the link fail state. It will then report the wire failure by generating a link fail interrupt to the host and will immediately stop the receive and transmit paths. These paths will not be enabled again until the Link Integrity Test ends successfully.

## 7.2 Media Access Control Operation

The LAN controller functions in a send and receive half- or full-duplex mode system. The 21041 functions in either transmit or receive in half-duplex mode or in both receive and transmit in loopback or full-duplex mode.

Before transmission, the 21041 checks that there is no competition for the network bus. In addition to listening for a clear line before transmitting, the 21041 handles collisions in a predetermined way. If two nodes attempt to transmit at the same time, the signals collide and the data on the line is garbled. The 21041 listens while it is transmitting, and it can detect a collision. If a collision is detected, the 21041 continues to transmit for a predetermined length of time, thus jamming the network, ensuring that all nodes have recognized the collision. The 21041 then delays its retransmission for a random time period determined by the truncated-binary backoff algorithm implemented in the 21041. This minimizes the possibility of a collision on retransmission.

## 7.2.1 Frame Format

The 21041 transmits or receives information in frames. It recognizes and transmits Ethernet and IEEE 802.3 frames.

### 7.2.1.1 Ethernet and IEEE 802.3 Frames

Ethernet is the generic name for the network type. An Ethernet frame has a minimum length of 64 bytes and a maximum length of 1518 bytes, exclusive of the preamble.

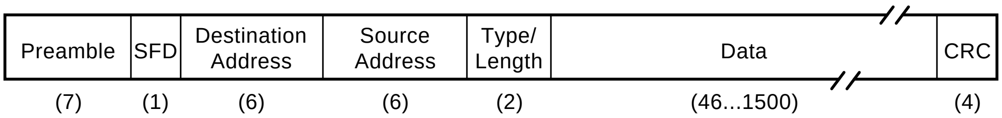
An Ethernet frame format consists of the following:

Preamble
Start frame delimiter (SFD)
Two address fields
Type or length field
Data field
Frame check sequence (FCS)

### 7.2.1.2 Frame Format Description

Figure 7–1 shows the Ethernet frame format.

**Figure 7–1 Ethernet Frame Format**




Numbers in parentheses indicates field length in bytes.

MLO-010295

The following list defines the byte field content for the Ethernet frame.

- Preamble—A 7-byte field of 56 alternating 1s and 0s, beginning with a 1.
- SFD (Start frame delimiter)—A 1-byte field that contains the value 10101011; the most significant bit is transmitted and received first.
- Destination Address—A 6-byte field that contains either a specific station address, the broadcast address, or a multicast (logical) address where this frame is directed.
- Source Address—A 6-byte field that contains the specific station address where this frame originated.
- Type/Length—A 2-byte field that indicates whether the frame is in IEEE 802.3 format or Ethernet format.

Table 7–2 lists the possible values for this field. The values are expressed in hexadecimal notation and the 2-byte field is displayed with a hyphen separating the 2 bytes. The byte on the left of the hyphen is the most significant byte (MSB) and is transmitted first.

**Table 7–2 Frame Format Table**

| Frame Format | Length or Type | Hexadecimal Value |
|---|---|---|
| IEEE 802.3 | Length field | 00-00 to 05-DC |
| Ethernet | Type field | 05-DD to FF-FF |

A field smaller or equal to 1500 (05-DC) is interpreted as a length field, which indicates the number of data bytes in the frame.

A field greater than 1500 is interpreted as a type field, which defines the type of protocol of the frame.

- Data—Consists of 46 to 1500 bytes of information that is fully transparent because any arbitrary sequence of bits can occur.

  A data field shorter than 46 bytes, which is specified by the length field, is allowed. If padding is enabled (TDES1<23>), it is added by the 21041 during transmission to fill the data field up to 46 bytes.

- FCS (Frame check sequence)—A 32-bit cyclic redundancy check (CRC) value that is computed as a function of the destination address field, source address field, type field, and data field. The FCS is appended to each transmitted frame and is used at reception to determine if the received frame is valid.

The CRC polynomial, as specified in the Ethernet specification, is as follows:

$$FCS(X) = X^{31} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} +$$

$$X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X^{1} + 1$$

The 32 bits of the CRC value are placed in the FCS field so that the $X^{31}$ term is the right-most bit of the first octet, and the $X^0$ term is the left-most bit of the last octet. The bits of the CRC are thus transmitted in the order $X^{31}$, $X^{30}$, ... , $X^1$, $X^0$.

### 7.2.2 Ethernet Reception Addressing

The 21041 can be set up to recognize any one of the Ethernet receive address groups described in Table 7–3. Each group is separate and distinct from the other groups.

**Table 7–3 Ethernet Receive Address Groups**

| Group | Description |
|---|---|
| 1 | 16 address perfect filtering<br><br>The 21041 provides support for the perfect filtering of up to 16 Ethernet physical or multicast addresses. Any mix of addresses may be used for this perfect filter function of the 21041. The 16 addresses are issued in setup frames to the 21041. |
| 2 | One physical address; unlimited multicast addresses imperfect filtering<br><br>The 21041 provides support for one, single physical address to be perfectly filtered with an unlimited number of multicast addresses to be imperfectly filtered. This case supports the needs of applications that require one, single physical address to be filtered as the station address, while enabling reception of more than 16 multicast addresses, without suffering the overhead of pass-all-multicast mode. The single physical address for perfect filtering, and a 512-bit mask for imperfect filtering using a hash algorithm, are issued in a setup frame to the 21041. When hash hits are detected, the 21041 delivers the received frame (Section 4.2.3). |
| 3 | Unlimited physical and multicast addresses imperfect filtering<br><br>The 21041 provides support for unlimited physical addresses to be imperfectly filtered with an unlimited number of multicast addresses to be imperfectly filtered as well. This case supports applications that require more than one physical address to be filtered as the station address, while enabling the reception of more than 16 multicast addresses, without suffering the overhead of pass-all-multicast mode. A 512-bit mask for imperfect filtering using a hash algorithm, is issued in a setup frame to the 21041. When hash hits are detected, the 21041 delivers the received frame (Section 4.2.3). |
| 4 | Promiscuous Ethernet reception<br><br>The 21041 provides support for reception of all frames on the network regardless of their destination. This function is controlled by a CSR bit. This group is typically used for network monitoring. |

(continued on next page)

**Table 7–3 (Cont.) Ethernet Receive Address Groups**

| Group | Description |
|---|---|
| 5 | 16 address perfect filtering and reception of all multicast Ethernet addresses<br>This group augments the receive address group 1 and also receives all frames on the Ethernet with a multicast address. |
| 6 | 16 address inverse filtering<br>In this mode, the 21041 applies the reverse filter of group 1. The 21041 provides support for the rejection of up to 16 Ethernet physical or multicast addresses. Any mix of addresses may be used for this filter function of the 21041. The 16 addresses are issued in setup frames to the 21041. |

### 7.2.3 Collision Detection and Implementation

The Ethernet CSMA/CD network access algorithm is implemented completely within the 21041. In addition to listening for a clear line before transmitting, the 21041 handles collisions in a predetermined way as defined by the standard. If two transmitters attempt to seize the line at the same time, a collision occurs and the data on the line is garbled. The 21041 listens while it transmits, and if a collision is detected, it continues to transmit for a predetermined length of time, thus *jamming* the network and ensuring that all nodes have recognized the collision.

The transmitting 21041 then delays retransmission for a random amount of time according to the Ethernet *truncated-binary-backoff* algorithm so that the colliding nodes do not try to repeatedly access the network at the same time. The 21041 makes up to 16 attempts to access the network before reporting an error due to excessive collisions (Table B–1).

### 7.2.4 Transmit Mode

In transmit mode, the 21041 initiates a DMA cycle to access data from a transmit buffer. It prefaces the data with a preamble and start frame delimiter (SFD) pattern, adds padding if needed, and then, if enabled, calculates and appends a 32-bit CRC.

After a frame is assembled, the 21041 waits for the internal transmit mechanism to allow transmission on the network, then serializes the data and outputs it to the SIA.

### 7.2.5 Receive Mode

In receive mode, the internal SIA sends decoded data and the clock to the internal receive mechanism. The data is then deserialized by the receive mechanism and fed into the internal FIFO.

As the data is received, the address is checked by the 21041 and a CRC is calculated and compared to the CRC checksum at the end of the frame. If the calculated CRC does not agree with the frame CRC, an error bit is set in the receive descriptor. The host processor is notified of all received frames, including those with CRC errors or excessive dribbling errors. These are called *runt* frames and are not delivered to the host unless the 21041 is programmed to do so. Both promiscuous mode (CSR6<6>) and pass bad frames (CSR6<3>) are set.

## 7.3 Detailed Transmission Operation

This section describes the transmission operation in detail as supported by the 21041. This description includes the specific control register definitions, setup frame definitions, and mechanism used by the host processor software to manipulate the transmit list (that is, the descriptors and buffers which can be found in Section 4.2).

### 7.3.1 Transmission Initiation

The host CPU initiates a transmission by storing the entire information content of the frame to be transmitted in one or more buffers in memory. The host processor software prepares a companion transmit descriptor, also in host memory, for the transmit buffer and signals the 21041 to take it. After the 21041 has been notified of this transmit list, the 21041 starts to move the data bytes from the host memory to the internal transmit FIFO.

When the transmit FIFO is adequately filled to the programmed threshold level, or when there is a full frame buffered into the transmit FIFO, the 21041 begins to encapsulate the frame.

The threshold level can be programmed with four quantities: 72, 96, 128, and 160 bytes (Table 3–39). The lower threshold is for low bus latency systems and the high threshold is for high bus latency systems.

The transmit encapsulation is performed by the transmit state machine, which delays the actual transmission of the data onto the network until the network has been idle for a minimum interpacket gap (IPG) time (9.6 microseconds).

## 7.3.2 Frame Encapsulation

The transmit data frame encapsulation stream consists of the following:

64 preamble bits
4 information fields
CRC

The CRC is computed in real time by the 21041 and, if enabled, is automatically appended to the frame at the end of the serial data. The preamble and CRC encapsulation support Ethernet frame format.

For outgoing frames, the following information is prepared by the host processor in the buffer memory before the host CPU initiates transmission:

Destination address
Source address
Type or length field
Data field

The 21041 encapsulates these fields into an Ethernet frame by inserting a preamble before the information fields and by appending both padding and an optional CRC after the information fields.

If the data field length is shorter than 46 bytes, the 21041 pads with 00s up to 46 bytes and also appends 4 bytes of CRC — regardless of the state of the add CRC disable (TDES1<26>) flag. This results in a frame of at least 64 bytes.

## 7.3.3 Initial Deferral

The 21041 constantly monitors the line and can initiate a transmission any time the host CPU requests it. Actual transmission of the data onto the network occurs only if the network has been idle for the 9.6-microsecond IPG time and any backoff time requirements have been satisfied.

The IPG time is divided into two parts: IPS1 and IPS2.

1. IPS1 time (6.0 microseconds)—The 21041 monitors the network for an idle state. If a carrier is sensed on the serial line during this time, the 21041 defers and waits until the line is idle again before restarting the IPS1 time count.
2. IPS2 time (3.6 microseconds)—The 21041 continues to count time even though a carrier has been sensed on the network, and thus forces collisions on the network. This enables all network stations to have access to the serial line.

### 7.3.4 Collision

A collision occurs when concurrent transmissions from two or more Ethernet nodes take place. The 21041 halts the transmission of the data bytes in the transmit FIFO and then transmits a jam pattern consisting of hexadecimal AAAAAAAA. At the end of the jam transmission, the 21041 begins the backoff wait period.

If the collision was detected during the preamble transmission, the jam pattern is transmitted after completing the preamble. This results in a minimum 96-bit fragment.

The 21041 scheduling of retransmission is determined by a controlled, random process called truncated binary exponential backoff. The delay is an integer multiple of slot times. The number of slot times of delay before the $n^{\text{th}}$ retransmission attempt is chosen as a uniformly distributed random integer $r$ in the range:

$0 \leq r < 2^k$
$k = \min(n, N)$ and $N = 10$

When 16 attempts have been made at transmission and all have been terminated by a collision, the 21041 sets an error status bit and, if enabled, issues an interrupt to the host.

**Note**

The jam pattern is a fixed pattern that is not compared with the actual frame CRC. This has the very low probability ($0.5^{32}$) of having a jam pattern equal to the CRC.

### 7.3.5 Terminating Transmission

A specific frame transmission is terminated by any of the following conditions:

- Normal—The frame has been transmitted successfully. When the last byte is serialized, the pad and CRC are optionally appended and transmitted, thus concluding frame transmission.
- Underflow—Transmit data is not ready when needed for transmission. The underflow status bits (TDES0<1> and CSR5<5>) are set, and the packet is terminated on the network with a bad CRC.
- Excessive collisions—If a collision occurs for the 15th consecutive retransmission attempt of the same frame, TDES0<8> is set.

- Jabber timer expired—If the timer expires (TDES0<14> sets) while transmission continues, the programmed interval transmission is cut off.
- Memory error—This generic error indicates either a host bus timeout or a host memory error.
- Late collision—If a collision occurs after the collision window (receiving at least 64 bytes), transmission is cut off and TDES0<9> sets.

At the completion of every frame transmission, status information about the frame is written into the transmit descriptor. Status information is written into CSR5 if an error occurs during the operation of the transmit machine itself.

### 7.3.6 Transmit Parameter Values

The transmit parameter values follow.

- Defer time: IPS1 + IPS2 = 96-bit-time = 9.6 microseconds for 10-megahertz serial bit rate
- IPS1 = 60-bit-time = 6.0 microseconds for 10-megahertz serial bit rate
- IPS2 = 36-bit-time = 3.6 microseconds for 10-megahertz serial bit rate
- Slot time interval = 512-bit-time = 51.2 microseconds for 10-megahertz serial bit rate (collision window)
- Network acquisition time = 512-bit-time = 51.2 microseconds for 10-megahertz serial bit rate, starting from the assertion of TXEN
- Transmission attempts limit: 16
- Backoff limit: 10
- Jabber timer: Default = 1.6 to 2 milliseconds, programmable range = 26 to 32 milliseconds.
- Append CRC on frame transmission: Programmable (yes or no).

## 7.4 Detailed Receiving Operation

This section describes the detailed receiving operation as supported by the 21041. This description includes the specific control register definitions, setup frame definitions, and mechanism used by the host processor software to manipulate the receive list (that is, the descriptors and buffers, which can be found in Section 4.2).

### 7.4.1 Initiating Reception

The 21041 continuously monitors the network when reception is enabled. When activity is recognized by a preamble being detected on the receive data lines, the 21041 synchronizes itself to the incoming data stream during the preamble, waits for the start frame delimiter (SFD), then examines the destination address field of the frame. Depending on the address match mode specified, the 21041 either recognizes the frame as being addressed to itself, or it discards it.

### 7.4.2 Preamble Processing

The preamble, as defined by Ethernet, can be up to 64 bits (8 bytes) long.

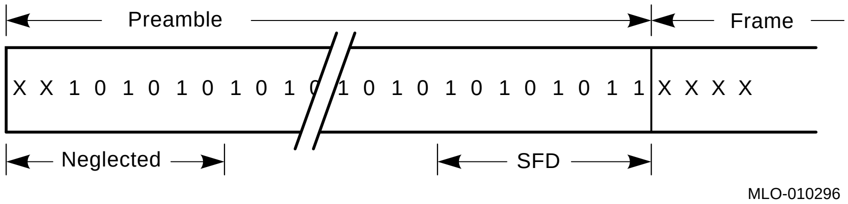
The 21041 allows any arbitrary preamble length, but needs at least 16 bits to recognize a preamble. Recognition occurs as follows:

1. The first 8 preamble bits are ignored.
2. The 21041 checks for the start frame delimiter (SFD) byte content of 10101011.

If the 21041 receives a 00 or a 11 after the first 8 preamble bits and before receiving the SFD, the reception of the current frame is aborted; the frame is not received, and the 21041 waits until the carrier drops and rises again, then begins monitoring the network for a new preamble.

Figure 7–2 shows the preamble recognition sequence bit fields.

**Figure 7–2 Preamble Recognition Sequence**




### 7.4.3 Address Matching

Ethernet addresses consist of two 6-byte fields, one field for the destination address and one for the source address. The first bit of the destination address signifies whether it is a physical address or a multicast address (Table 7–4).

**Table 7–4 Destination Address Bit 1**

| Bit 1 | Address |
|---|---|
| 0 | Station address (physical) |
| 1 | Multicast address |

The 21041 filters the frame based on the Ethernet receive address group (Section 7.2.2) filtering mode that has been enabled.

If the frame address passes the filter, the 21041 removes the preamble and delivers the frame to the host processor memory. If, however, the address does not pass the filter when the mismatch is recognized, the 21041 terminates its reception. In this case, no data is sent to the host memory nor is any receive buffer consumed.

### 7.4.4 Frame Decapsulation

The 21041 checks the CRC bytes of all received frames before releasing the frame along with the CRC to the host processor.

### 7.4.5 Terminating Reception

Reception of a specific frame is terminated when any of the following conditions occur:

- Normal termination—The carrier sense line becomes inactive, indicating that traffic is no longer present on the Ethernet cable.
- Overflow—The receive DMA cannot empty the receive FIFO into host processor memory as rapidly as it is filled, and an error occurs as frame data is lost. The overflow status bit (RDES0<0>) is set.
- Watchdog timer expired—The timer expires (CSR5<9> and RDES0<4> both set) while reception is still in process, and reception is cut off.
- Collision—If a late collision occurs after the reception of 64 bytes of the packet, the collision seen status bit (RDES0<6>) is set.

### 7.4.6 Frame Reception Conditions

When reception terminates, the 21041 determines the status of the received frame and loads it into the receive status word in the buffer descriptor. An interrupt is issued if enabled. The 21041 may report the following conditions at the end of frame reception:

- Overflow—The 21041 receive FIFO overflowed.
- CRC error—The 32-bit CRC transmitted with the frame did not match the CRC calculated upon reception. The CRC check is always executed and is independent of any other errors.
- Dribbling bits error—This indicates the frame did not end on a byte boundary. The 21041 signals a dribbling bits error only if it detects more than two dribbling bits. Only *whole* bytes are run through the CRC check. This means that although up to seven dribbling bits may have occurred and a framing error was signaled, the frame might nevertheless have been received correctly.
- Alignment error—A CRC error and dribbling bit error occur together. This means that the frame did not contain an integral number of bytes and the CRC check failed.
- Frame too short (runt frame)—A frame containing less than 64 bytes was received (including CRC). Reception of runt frames is optionally selectable. The 21041 defaults to inhibit reception of runts.
- Frame too long—A frame containing more than 1500 bytes was received. Reception of frames too long completes with an error indication.
- Collision seen—A frame collision occurred after the 64 bytes following the start frame delimiter (SFD) were received. Reception of such frames is completed and an error bit is set in the descriptor.
- Descriptor error—An error was found in one of the receive descriptors, which disabled the correct reception of an incoming frame.

### 7.4.7 Capture Effect

The 21041 provides a complete solution for the capture effect on the network. (See Section 3.3.7.) Capture effect is defined as a station that captures a channel for an unfair amount of time, transmitting its packets back-to-back, while another station continues to back off and is unable to transmit its waiting packets. In the next subsections, the capture effect problem and its resolution are explained.

#### 7.4.7.1 What is Capture Effect?

Consider two stations on the line, station A and station B. Each station has a significant amount of data ready to transmit. Each station is able to satisfy the minimum IPG rules (both from transmit-to-transmit and from receive-to-transmit). The following steps show how station A captures the line (Table 7–5).

1. Station A (with data A1) and station B (with data B1) both attempt to transmit simultaneously within a slot time of 51.2 microseconds. Each station has an initial collision count set to 0.
2. The stations experience a collision. Both stations increment their collision count to 1.
3. Each station picks a backoff time value which is uniformly distributed from 0 to (2n)–1 slots. In this example, station B selects a backoff of 1 (a 50% probability), and station A selects a backoff of 0.
4. Station A successfully transmits its A1 data packet. Station B waits for data A1 to be transmitted before attempting to retransmit data B1.
5. Collision count at station B remains at 1, while collision count at station A is reset to 0.
6. If station A has another packet (data A2) ready to transmit while station B still wants to transmit its packet (data B1), the stations both contend for the line again.
7. If these stations collide, the backoff value available for station A is 0 or 1 slots. The backoff value available for station B is 0, 1, 2, or 3 slots because the collision count is now at 2 (station A's collision count is at 1). Station A is more likely to succeed and transmit data A2, while data B1 from station B begins the deferral of completing its backoff interval.
8. It is possible, with this type of behavior between stations, that in the 2-node Ethernet, a station can capture the channel for an unfair amount of time. One station can transmit a significant number of packets back-to-back, while the other station continues to backoff further and further.

9. This process could continue until station B reaches excessive collision while attempting to transmit data B1. At this time, station B has the same probability to win the line for the next transmit data as Station A does.

**Note**

If station A completes the transmitting of a stream of packets during this type of capture, and station B is still in backoff, potentially for a long time, the line is idle for this period of time.

**Table 7–5 Capture-Effect Sequence**

| Station A | Line | Station B | Collision Count A | Collision Count B |
|---|---|---|---|---|
| Transmit packet A1 | Collision | Transmit packet B1 | 0 | 0 |
| Backoff 0, 1 | | Backoff 0, 1 | 1 | 1 |
| Transmit packet A1 | Packet A1 | Backoff | 0 | 1 |
| Transmit packet A2 | Collision | Transmit packet B1 | 0 | 1 |
| Backoff 0, 1 | | Backoff 0, 1, 2, 3 | 1 | 2 |
| Transmit packet A2 | Packet A2 | Backoff | 0 | 2 |
| Transmit packet A3 | Collision | Transmit packet B1 | 0 | 2 |
| Backoff 0, 1 | | Backoff 0, 1, 2, ... 7 | 1 | 3 |

#### 7.4.7.2 Resolving Capture Effect

The 21041 generally resolves the capture effect by having the station use, after a successful transmission of a frame by a station, a 2-0 backoff algorithm on the next transmit frame. If the station senses a frame on the network before it attempts to transmit the next frame, regardless of whether the sensed frame destination address matches the station's source address, the station returns to use the standard truncated, binary exponential backoff algorithm (Section 7.3.4).

When the station executes the 2-0 backoff algorithm, it always waits for a 2-slot period on the first collision, and for a 0-slot period on the second collision. For retransmission attempts greater than 2, it uses the standard truncated, binary exponential backoff algorithm.

A programmable bit, capture effect enable (CSR6<17>), is implemented to activate the capture effect resolution.

Table 7–6 summarizes the 2-0 backoff algorithm.

**Table 7–6 The 2-0 Backoff Algorithm**

| Retransmission Attempts | Backoff Period (Number of Slot Times) |
|---|---|
| n = 1 | Backoff = 2 slots |
| n = 2 | Backoff = 0 slots |
| n = 3–15 | Backoff = $0 \leq r < 2^k$ |
| | k = min (n, N) and N = 10<br>r = uniformly distributed random integer |

#### 7.4.7.3 Enhanced Resolution for Capture Effect

The 21041 offers an enhanced resolution for capture effect. The enhancement is made by incorporating a stopped backoff algorithm (with the 2-0 backoff algorithm) to reduce collision while maintaining the key properties of the 2-0 backoff algorithm.

When the enhanced resolution for the capture effect bit is set (CSR6<31>), the 21041 activates the stopped backoff algorithm as follows: in a back-to-back transmit, while in backoff after the first collision (n=1, where n is the retransmission attempts), the 21041 stops its backoff timer for the duration when the channel is busy. It continues its backoff timer when the channel is idle. For any other collision cases, the backoff timer is not stopped.

# 8
# Boot ROM and Serial ROM Interfaces

This chapter describes the interface to the boot ROM and serial ROM, the connection, and the read and write cycles.

## 8.1 Overview

The 21041 provides a boot ROM interface that may be optionally used on the adapter. The boot ROM (expansion ROM) contains a code that can be executed for device-specific initialization and, possibly, a system boot function. During machine boot, the BIOS looks for bootable devices by searching a specific signature (55AA). Once found, the BIOS copies the code from the boot ROM to a shadow RAM in the host memory and executes the code from the RAM.

The boot ROM interface supports:

– 5-volt or 12-volt FLASH memory for code upgrade

– 120 ns EEPROM or faster

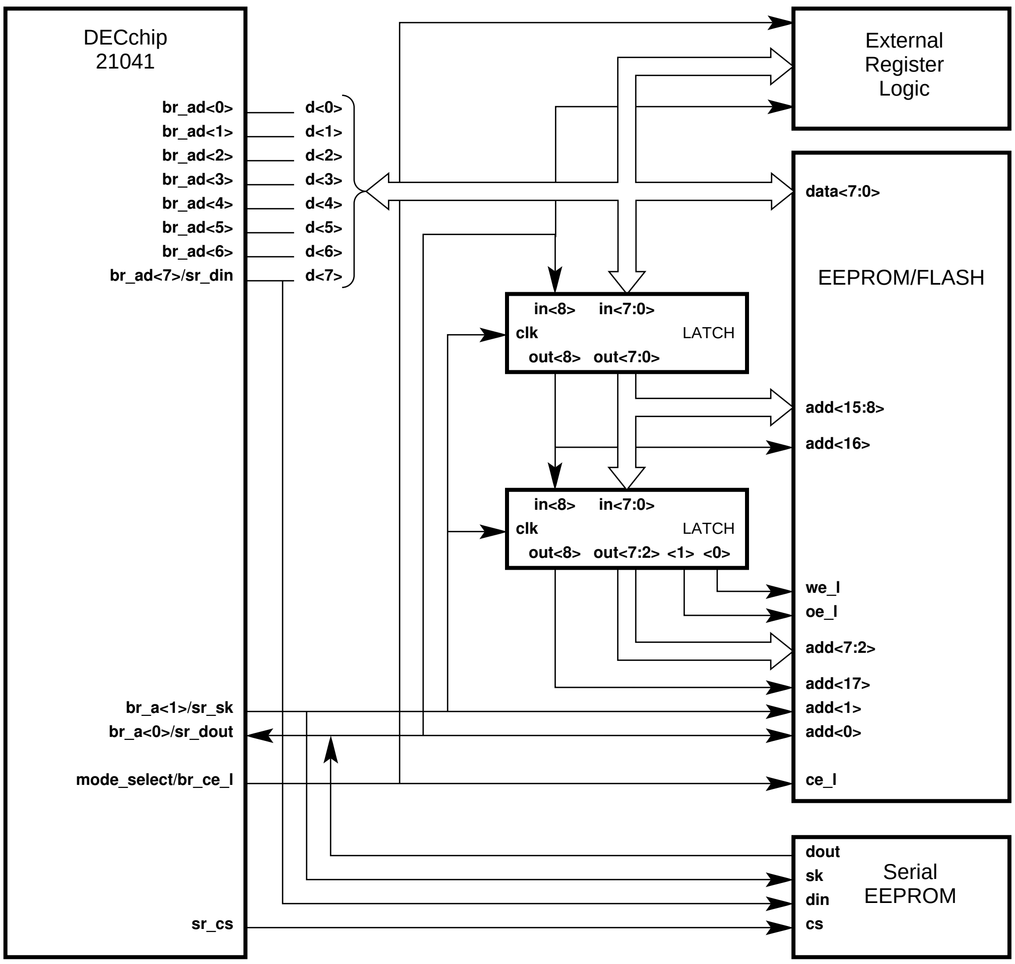
– Up to 256K-byte address space

The 21041 provides a software-controlled, serial port interface suitable for MICROWIRE and other common serial ROM devices. The serial ROM contains the IEEE address and, optionally, other system parameters. In addition, the 21041 provides the ability to perform read and write operations with an external 8-bit register.

## 8.2 Boot ROM and Serial ROM Connection

Figure 8–1 shows the connection of a 256K-byte boot ROM and the serial ROM. The two 9-bit edge trigger latches are used to latch the boot ROM addresses <17:2> and the oe_l and we_l control signals.

**Figure 8–1 Boot ROM, Serial ROM, and External Register Connection**




LJ-03979.AI

## 8.3 Boot ROM Operation

Access to the boot ROM is done in two ways:

1. Byte access (read/write) using CSR9 and CSR10.
2. DWORD (32-bit) read access from the PCI expansion ROM address space.

The following sections describe these accesses. For each, the boot ROM must be set to the desired mode (read or write) prior to the actual access for the read or write. For additional information about how this is done, refer to the specific ROM device documentation.

Any mixture between byte access and DWORD access is allowed providing that byte access followed by DWORD access will be separated by at least 15 PCI clock cycles.

**Note**

CSR13 must be 0 before doing any boot ROM access.

---

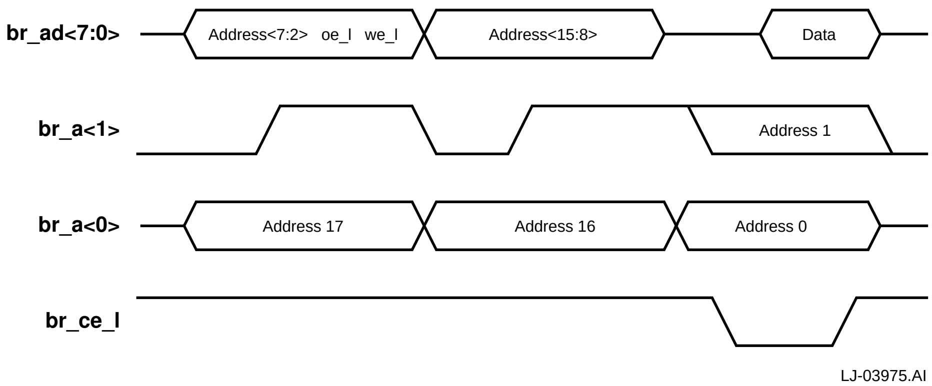
### 8.3.1 Byte Read

Figure 8–2 shows the 21041 byte read cycle. It is executed as follows:

1. The host initiates a byte read cycle to the boot ROM by writing the boot ROM offset to CSR10 and setting a read command in CSR9 (CSR9<14> and CSR9<12> = 1).
2. The 21041 drives the boot ROM address bits <7:2> and the signals oe_l and we_l on the br_ad lines, drives address bit 17 on the br_a<0> line, and sets br_a<1>. br_a<1> is used as a latch_enable to latch the address, oe_l, and we_l in the upper edge trigger latch.
3. The 21041 clears br_a<1>.
4. The 21041 drives the boot ROM address bits <15:8> on the br_ad lines, drives address bit 16 on the br_a<0> line, and sets br_a<1>. Address bits <16:8> are latched in the upper edge trigger latch while the previous address bits (17, <7:2>) and the control signals (oe_l and we_l) are latched in the lower edge trigger latch.
5. The 21041 drives address bits <1:0> on br_a<1> and br_a<0> respectively and asserts the br_ce_l pin.
6. In response, the boot ROM drives the data on the br_ad lines.

7. The 21041 terminates the byte read cycle by sampling the data, placing it in CSR9<7:0>, and deasserting the br_ce_l signal.
8. The driver can read CSR9 for the data after at least 20 PCI clock cycles passed since this CSR was previously written. Note that the results of trying to read the data earlier are unpredictable.

**Figure 8–2 Boot ROM Byte Read Cycle**




## 8.3.2 Byte Write

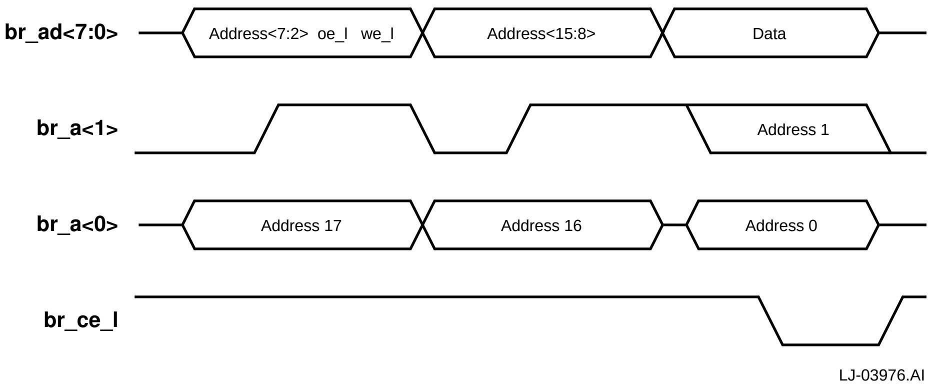
Before performing a write operation, all the boot ROM entries must be 1. This is achieved by using the erase command.

Figure 8–3 shows the 21041 byte write cycle. It is executed as follows:

1. The host initiates a byte write cycle to the boot ROM by writing the boot ROM offset to CSR10, setting a write command in CSR9 (CSR9<13> and CSR9<12> = 1), and writing the data to CSR9<7:0>.
2. The 21041 drives the boot ROM address bits <7:2> and the signals oe_l and we_l on the br_ad lines, drives address bit 17 on the br_a<0> line, and sets br_a<1>. br_a<1> is used as a latch_enable to latch the address, oe_l, and we_l in the upper edge trigger latch.
3. The 21041 clears br_a<1>.

4. The 21041 drives the boot ROM address bits <15:8> on the br_ad lines, drives address bit 16 on the br_a<0> line, and sets br_a<1>. Address bits <16:8> are latched in the upper edge trigger latch while the previous address bits (17, <7:2>) and the control signals (oe_l and we_l) are latched in the lower edge trigger latch.
5. The 21041 drives address bits <1:0> on br_a<1> and br_a<0> respectively, drives the data on the br_ad lines, and asserts the br_ce_l pin.
6. The boot ROM samples the data.
7. The 21041 terminates the byte write cycle by deasserting the br_ce_l signal.

**Figure 8–3 Boot ROM Byte Write Cycle**



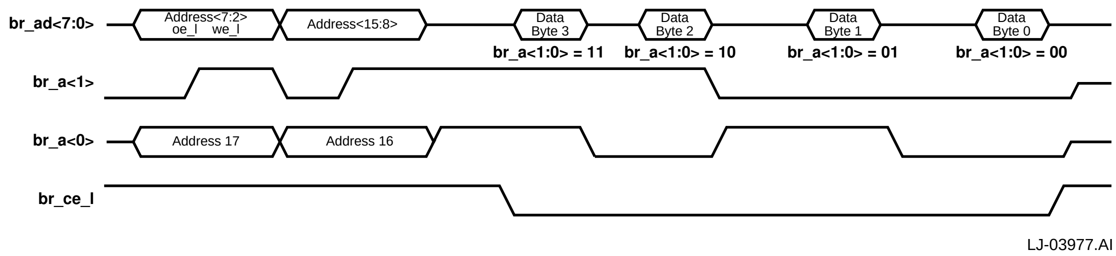

### 8.3.3 DWORD Read

Figure 8–4 shows the DWORD read cycle. The host initiates a DWORD read cycle by executing a typical slave read cycle to the expansion ROM address space. The ad lines contain the expansion ROM address (base address and offset). Prior to the assertion of the trdy_l signal, the 21041 takes the following steps:

1. The 21041 drives the boot ROM address bits <7:2> and the control signals oe_l and we_l on the br_ad lines, drives address bit 17 on the br_a<0> line, and sets br_a<1>. br_a<1> is used as a latch_enable to latch the address, oe_l, and we_l in the upper edge trigger latch.
2. The 21041 clears br_a<1>.

3. The 21041 drives the boot ROM address bits <15:8> on the br_ad lines, drives address bit 16 on the br_ad<0> line, and sets br_a<1>. Address bits <16:8> are latched in the upper edge trigger latch while the previous address bits (17, <7:2>) and the control signals (oe_l and we_l) are latched in the lower edge trigger latch.
4. The 21041 remains br_a<1> high, drives br_a<0> to high, and asserts the br_ce_l pin.
5. In response, the boot ROM drives the data on the br_ad lines (byte 3).
6. The 21041 samples the data (byte 3).
7. The 21041 remains br_a<1> high, drives br_a<0> to low, and asserts the br_ce_l pin.
8. In response, the boot ROM drives the data on the br_ad lines (byte 2).
9. The 21041 samples the data (byte 2).
10. The 21041 drives br_a<1> to low, drives br_a<0> high, and asserts the br_ce_l pin.
11. In response, the boot ROM drives the data on the br_ad lines (byte 1).
12. The 21041 samples the data (byte 1).
13. The 21041 remains br_a<1> low, drives br_a<0> to low, and asserts the br_ce_l pin.
14. In response, the boot ROM drives the data on the br_ad lines (byte 0).
15. The 21041 samples the data and deasserts the br_ce_l signal.
16. The 21041 assembles the 4 bytes, drives the data on the ad lines, and asserts trdy_l.

**Figure 8–4 Boot ROM DWORD Read Cycle**




## 8.4 Serial ROM Operation

The serial ROM pins are fully software-driven. All EEPROM pins (sr_cs, sr_sk, sr_din, sr_dout) are connected directly to the CSR9<3:0> bits. The software handles all EEPROM access sequences and timing.

The serial ROM operations include the following:

- Read operation
- Write operation

The following sections describe these operations. Additional operations, such as erase EEPROM, are also supported and handled in the same manner.

### 8.4.1 Read Operation

Each read operation consists of the following three phases:

1. Command phase (3 bit–110#2)
2. Address phase (6 bit for 256–1024 bit ROMs, 8 bit for 2048–4096 bit ROMs)
3. Data phase (16 bit)

Figure 8–5 shows an example of a typical read cycle using the MICROWIRE serial ROM. Figures 8–6 and 8–7 are a flowchart that shows the actions the driver must take to execute such a read cycle, and it provides the minimum time, in nanoseconds (ns), that the driver must wait before moving to the next action.

**Figure 8–5 MICROWIRE Serial ROM Read Cycle**

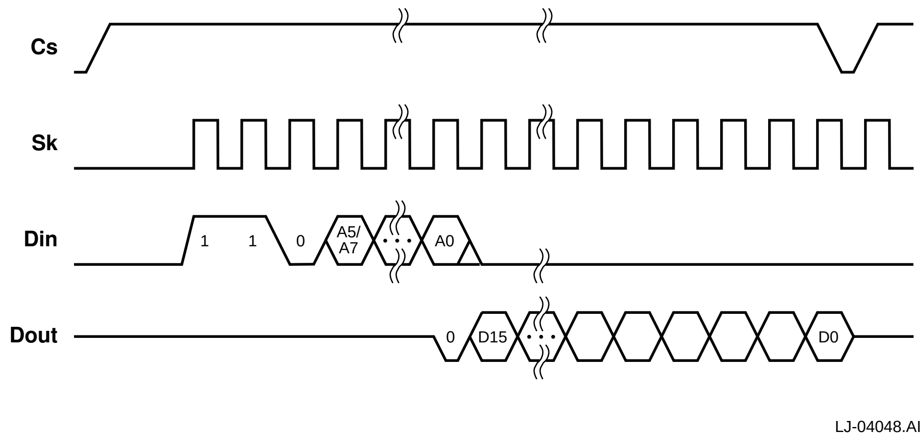


**Figure 8–6 MICROWIRE Serial ROM Read Cycle Flowchart**

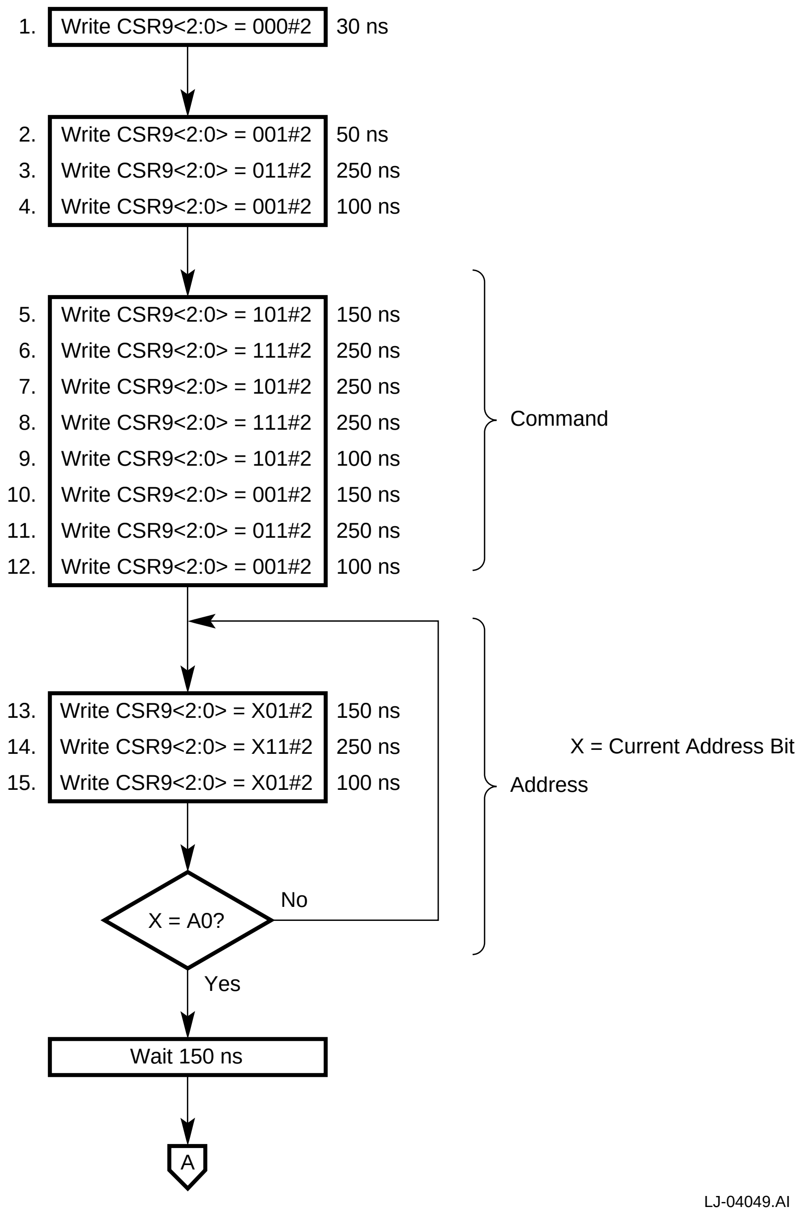
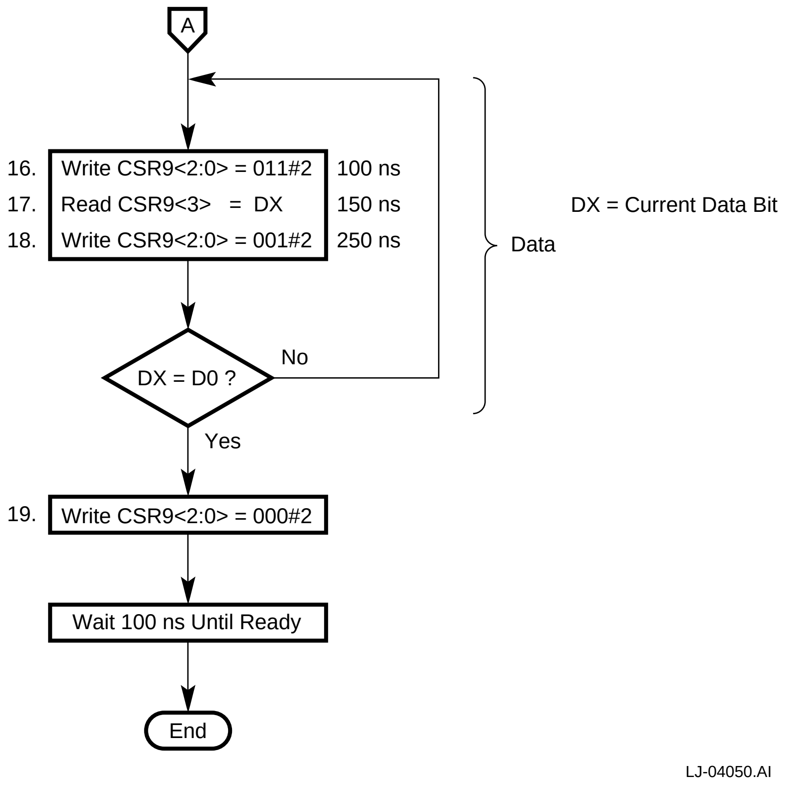


**Figure 8–7 MICROWIRE Serial ROM Read Cycle Flowchart (continued)**



## 8.4.2 Write Operation

Each write operation consists of the following four phases:

1. Command phase (3 bit—101#2)
2. Address phase (6 bit for 256–1024 bit ROMs, 8 bit for 2048–4096 bit ROMs)
3. Data phase (16 bit)
4. Busy—the serial ROM has not completed the current write operation. Reading dout as 1 indicates that it is ready again.

Figure 8–8 shows an example of a typical write cycle using the MICROWIRE serial ROM. Figures 8–9 and 8–10 are a flowchart that shows the actions the driver must take to execute such a write cycle, and it provides the minimum time, in nanoseconds (ns), that the driver must wait before moving to the next action.

**Figure 8–8 MICROWIRE Serial ROM Write Cycle**

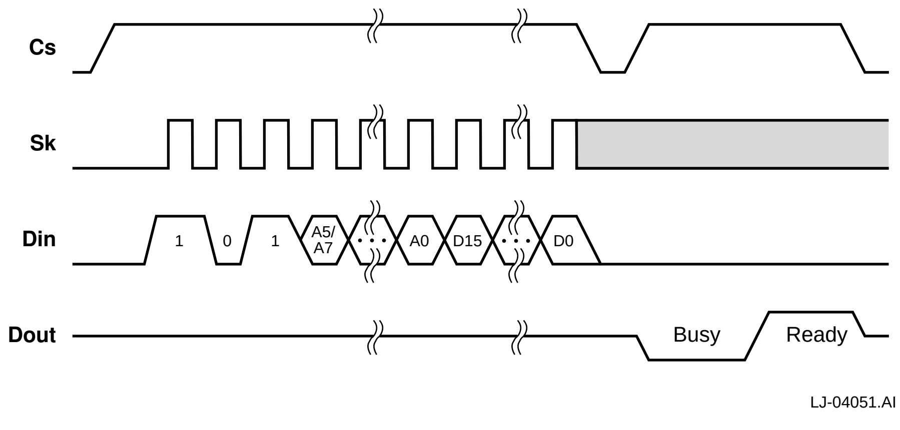


**Figure 8–9 MICROWIRE Serial ROM Write Cycle Flowchart**

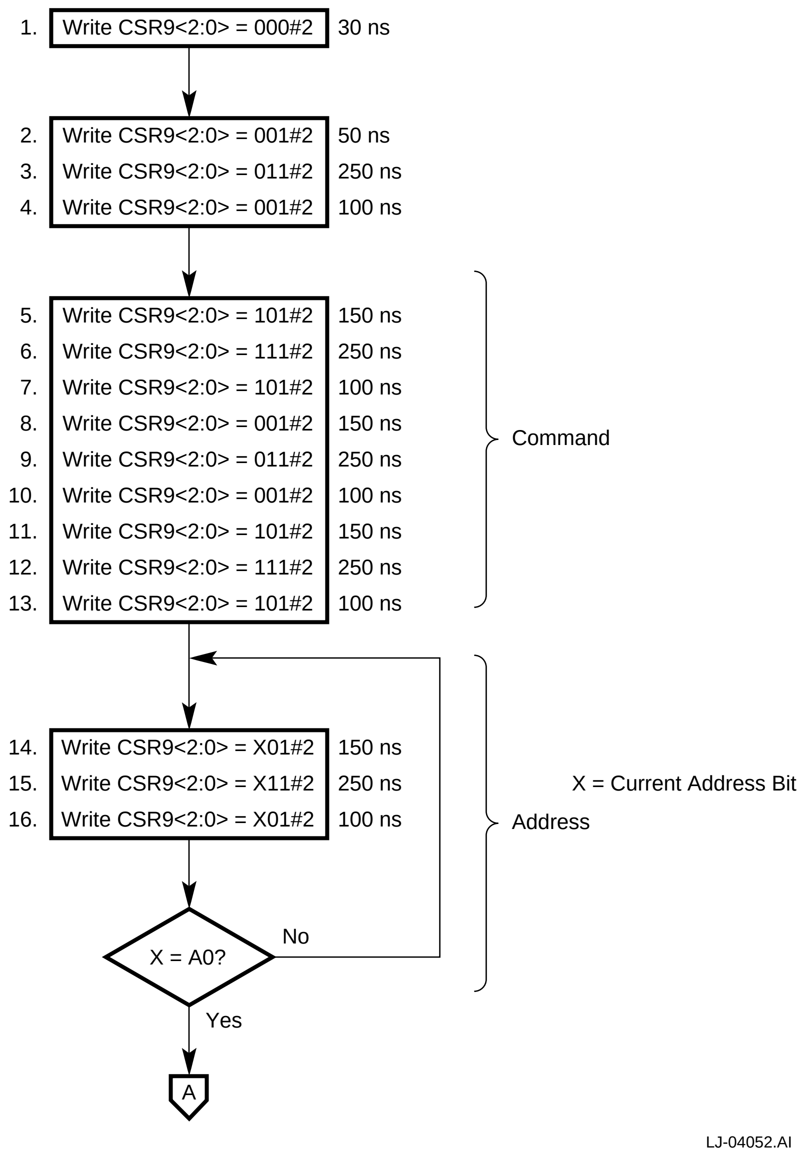
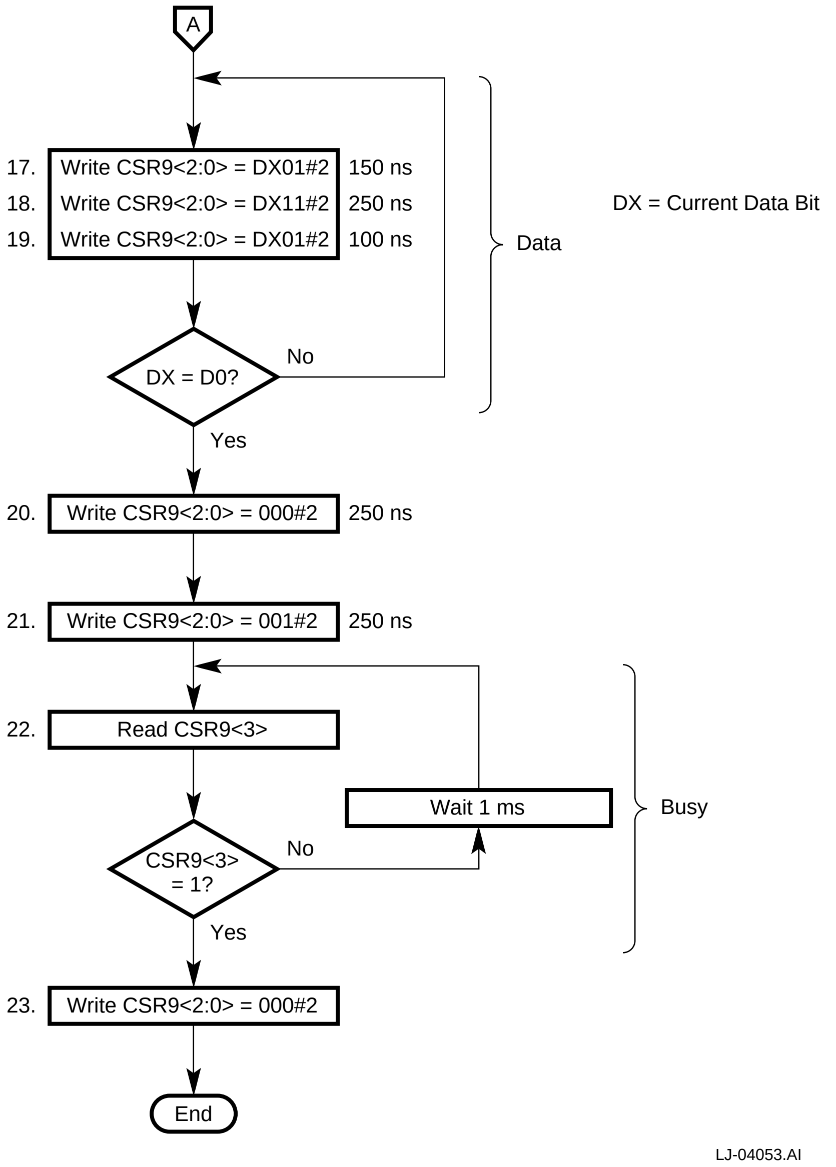


**Figure 8–10 MICROWIRE Serial ROM Write Cycle Flowchart (continued)**



## 8.5 External Register Operation

The 21041 provides the ability to connect an external 8-bit register to the boot ROM port. Figure 8–1 illustrates the signals that take part in this connection. For a detailed description of external register connection, refer to *Connecting the DECchip 21041 PCI Ethernet LAN Controller to the Network: An Application Note*.

**Note**

CSR10 must be 0 before any external register access is done.

---

To read from the external register, the driver should set the read command (CSR9<14>) and select the external register (CSR9<10>=1). The 21041 performs the same steps as described in Section 8.3.1. The only differences are that now the 21041 drives 1 on both the we_l and oe_l boot ROM inputs and drives 0 on br_a<0>/sr_dout. This, together with the assertion of ce_l, performs the actual read operation. The data is sampled by the 21041 and placed in CSR9<7:0>.

To write to the external register, the driver should set the write command (CSR<13>), select the external register (CSR9<10>=1), and write the data to CSR9<7:0>. The 21041 performs the same steps as described in Section 8.3.2. The only differences are that now the 21041 drives 1 on both the we_l and oe_l boot ROM inputs and drives 1 on br_a<0>/sr_dout. This, together with the assertion of ce_l, performs the actual write operation.

# A
# Joint Test Action Group Test Logic

This appendix describes the joint test action group (JTAG) test logic and the associated registers (instruction, bypass, and boundary scan).

**Note**

To understand the description of the 21041 JTAG test logic in this section, the system designer should be familiar with the IEEE 1149.1 standard.

---

## A.1 General Description

JTAG test logic supports testing, observing, and modifying circuit activity during the components' normal operation. As a PCI device, the 21041 supports the IEEE standard 1149.1 *test access port and boundary scan architecture*. The IEEE 1149.1 standard specifies the rules and permissions that govern the design of the 21041 JTAG test logic support. Inclusion of JTAG test logic allows boundary scan to be used to test both the device and the board where it is installed. The JTAG test logic consists of the following four pins to serially interface within the 21041 (Table 2–1).

TCK—JTAG clock
TDI—Test data and instructions in
TDO—Test data and instructions out
TMS—Test mode select

**Note**

If JTAG test logic is not implemented, the TCK pin should be connected to ground. TDI, TMS, and TDO should remain unconnected (TDI and TMS are internally pulled up).

---

These test pins operate in the same electrical environment as the 21041 PCI I/O buffers.

The system vendor is responsible for the design and operation of the 1149.1 serial chains (rings) required in the system. Typically, an 1149.1 ring is created by connecting one device's TDO pin to another device's TDI pin to create a serial chain of devices. In this application, the 21041 receives the same TCK and TMS signals as the other devices. The entire 1149.1 ring is connected to either a motherboard test connector for test purposes or to a resident 1149.1 controller.

## A.2 Registers

In JTAG test logic design, three registers are implemented through all the 21041 pads:

Instruction register
Bypass register
Boundary scan register

### A.2.1 Instruction Register

The 21041 JTAG test logic instruction register is a 3-bit (IR<2:0>) scan register that is used to direct the JTAG machine to the appropriate operating JTAG mode (Table A–1). Its contents are interpreted as test instructions. The test instructions select the boundary scan registers for serial transfer of test data by using the TDI and TDO pins. These instructions also control the operation of the selected test features.

**Table A–1 Instruction Register**

| IR<2> | IR<1> | IR<0> | Description |
|---|---|---|---|
| 0 | 0 | 0 | Extest mode (mandatory instruction) allows testing of the 21041 board-level interconnections. Test data is shifted into the boundary scan register of the 21041 and then transferred in parallel to the output pins. |
| 0 | 0 | 1 | Sample-preload mode (mandatory instruction) allows the 21041 JTAG boundary scan register to be initialized prior to selecting other instructions such as EXTEST. It is also possible to capture data at system pins while the system is running, and to shift that data out for examination. |
| 0 | 1 | 0 | Reserved. |
| 0 | 1 | 1 | Reserved. |
| 1 | 0 | 0 | Reserved. |
| 1 | 0 | 1 | Tristate mode (optional instruction) places all 21041 external pads in an inactive drive state (high impedance). When this occurs, all internal clocks are frozen, all external pads enter tristate, and internal phase-locked loops reduce their power consumption. All the SIA transceivers continue to operate normally without any power reduction. |
| 1 | 1 | 0 | Continuity mode (optional instruction) allows the 21041 continuity test while in production. |
| 1 | 1 | 1 | Bypass mode (mandatory instruction) allows the test features on the 21041 JTAG test logic to be bypassed. This instruction selects the bypass register to be connected between TDI and TDO.<br><br>When the bypass mode is selected, the operation of the test logic has no effect on the operation of the system logic.<br><br>Bypass mode is selected automatically, when power is applied. |

## A.2.2 Bypass Register

The bypass register is a 1-bit shift register that provides a single-bit serial connection between the TDI and TDO pins when either no other test data register in the 21041 JTAG test logic registers is selected, or the test logic in the 21041 JTAG is bypassed. When power is applied, JTAG test logic resets and then is set to bypass mode.

### A.2.3 Boundary Scan Register

The JTAG boundary scan register consists of cells located at the PCI and the boot ROM (EXT_SIA) pins of the 21041. This register provides an interconnections test. It also provides additional control and observation of the 21041 pins during the testing phases. For example, the 21041 boundary scan register can observe the output enable control signals of the I/O pads; for example, ad_oe, cbe_oe, and so on. When these signals are programmed to be 1 during extest mode, data is applied to the output from the selected pins.

The following listing contains boundary scan register pads order.

**Note**

There are no boundary scan register pads on the TP or AUI ports.

```
-> tdi        -> br_oel       -> br_ad<0>    -> br_oe2      -> br_ad<1>
-> br_ad<2>   -> br_ad<3>     -> br_oe3      -> br_ad<4>    -> br_ad<5>
-> br_ad<6>   -> br_oe4       -> br_ad<7>/   -> aui_bnc_oe  -> aui_bnc
                                 sr_din
-> mode_oe    -> mode_select/ -> sr_cs       -> br_a<1>/    -> sdo_oe
                 br_ce_l                        sr_sk
-> br_a<0>/   -> ad<0>        -> ad<1>       -> ad<2>       -> ad<3>
   sr_dout
-> ad<4>      -> ad<5>        -> ad<6>       -> ad<7>       -> c_be_l<0>
-> ad<8>      -> ad<9>        -> ad<10>      -> ad<11>      -> ad<12>
-> ad<13>     -> ad<14>       -> ad<15>      -> c_be_l<1>   -> par_oe
-> par        -> serr_l       -> perr_oe     -> perr_l      -> stop_oe
-> stop_l     -> ad_oe        -> cbe_oe      -> devsel_oe   -> devsel_l
-> trdy_oe    -> trdy_l       -> irdy_oe     -> irdy_l      -> frame_oe
-> frame_l    -> c_be_l<2>    -> ad<16>      -> ad<17>      -> ad<18>
-> ad<19>     -> ad<20>       -> ad<21>      -> ad<22>      -> ad<23>
-> idsel      -> c_be_l<3>    -> ad<24>      -> ad<25>      -> ad<26>
-> ad<27>     -> ad<28>       -> ad<29>      -> ad<30>      -> ad<31>
-> req_l      -> gnt_l        -> clk         -> rst_l       -> int_l
-> tdo
```

Table A–2 lists the boundary scan register controls.

**Table A–2 Boundary Scan Register Controls**

| Signal | Pin Control |
|---|---|
| ad_oe | ad<31:0> |
| cbe_oe | c_be_l<3:0> |
| frame_oe | frame_l |
| devsel_oe | devsel_l |
| stop_oe | stop_l |
| par_oe | par_l |
| irdy_oe | irdy_l |
| trdy_oe | trdy_l |
| perr_oe | perr_l |
| br_oe1 | br_ad<0>, br_ad<2> |
| br_oe2 | br_ad<1>, br_ad<3> |
| br_oe3 | br_ad<6:4> |
| br_oe4 | br_ad<7>/sr_din |
| aui_bnc_oe | aui_bnc |
| mode_oe | mode_select/br_ce_l |
| sdo_oe | br_a<0>/sr_dout |

## A.2.4 Test Access Port Controller

The test access port (TAP) controller interprets IEEE P1149.1 protocols received on the TMS pin. The TAP controller generates clocks and control signals to control the operation of the test logic. The TAP controller consists of a state machine and control dispatch logic. The 21041 fully implements the TAP state machine as described in the IEEE P1149.1 standard.

# B

# DNA CSMA/CD Counters and Events Support

This appendix describes the 21041 features that support the driver in implementing and reporting the specified counters and events.[1] CSMA/CD[2] specified events can be reported by the driver based on these features. Table B–1 lists the counters and features.

**Table B–1 CSMA/CD Counters**

| Counter | 21041 Feature |
|---|---|
| Time since creation counter | Supported by the host driver. |
| Bytes received | Driver must add the frame length (RDES0<30:16>) fields of all successfully received frames. |
| Bytes sent | Driver must add the buffer 1 size (TDES1<10:0>) and buffer 2 size (TDES1<21:11>) fields of all successfully transmitted buffers. |
| Frames received | Driver must count the successfully received frames in the receive descriptor list. |
| Frames sent | Driver must count the successfully transmitted frames in the transmit descriptor list. |
| Multicast bytes received | Driver must add the frame length (RDES0<30:16>) fields of all successfully received frames with multicast frame (RDES0<10>) set. |
| Multicast frames received | Driver must count the successfully received frames with multicast frame (RDES<10>) set. |

(continued on next page)

[1] As specified in the *DNA Maintenance Operations Protocol (MOP) Functional Specification*, Version T.4.0.0, 28 January 1988.

[2] Carrier-sense multiple access with collision detection

**Table B–1 (Cont.) CSMA/CD Counters**

| Counter | 21041 Feature |
|---|---|
| Frames sent, initially deferred | Driver must count the successfully transmitted frames where deferred (TDES0<0>) is set. |
| Frames sent, single collision | Driver must count the successfully transmitted frames where collision count (TDES0<6:3>) is equal to 1. |
| Frames sent, multiple collisions | Driver must count the successfully transmitted frames where collision count (TDES0<6:3>) is greater than 1. |
| Send failure, excessive collisions | Driver must count the transmit descriptors where the excessive collisions (TDES0<8>) bit is set. |
| Send failure, carrier check failed | Driver must count the transmit descriptors where both late collision (TDES0<9>) and loss of carrier (TDES0<11>) are set. |
| Send failure, short circuit | There were two successive transmit descriptors where the no_carrier flag (TDES0<10>) is set. This indicates a short circuit. |
| Send failure, open circuit | There were two successive transmit descriptors where the excessive_collisions flag (TDES0<8>) is set. This indicates an open circuit. |
| Send failure, remote failure to defer | Flagged as a late collision (TDES0<9>) in the transmit descriptors. |
| Receive failure, block check error | Driver must count the receive descriptors where CRC error (RDES0<1>) is set and dribbling bit (RDES0<2>) is cleared. |
| Receive failure, framing error | Driver must count the receive descriptors where both CRC error (RDES0<1>) and dribbling bit (RDES0<2>) are set. |
| Receive failure, frame too long | Driver must count the receive descriptors where frame too long (RDES0<7>) is set. |
| Unrecognized frame destination | Not applicable. |
| Data overrun | Driver must count the receive descriptors where overflow (RDES0<0>) is set. |
| System buffer unavailable | Reported in the missed frame counter CSR8<15:0> (Section 3.3.9). |
| User buffer unavailable | Kept by the driver. |

(continued on next page)

**Table B–1 (Cont.) CSMA/CD Counters**

| Counter | 21041 Feature |
|---|---|
| Collision detect check failed | Driver must count the transmit descriptors where heartbeat fail (TDES0<7>) is set. |

# C

# Hash C Routine

This appendix provides two C routine examples that generate the hash index for a given Ethernet address. The first example is for little endian architecture and the other example is for big endian architecture.

The bit position in the hash table is taken from the CRC32 checksum derived from the first 6 bytes.

## C.1 Hash C Routine for Little Endian Architecture

```
#define CRC32_POLY    0xEDB88320UL      / *CRC-32 Poly -- Little Endian */
#define HASH_BITS     9                 / *Number of bits in hash */

unsigned
crc32_mchash(
    unsigned char *mcs)
{
    u_int idx, bit, data, crc = 0xFFFFFFFFUL;

    for (idx = 0; idx < 6; idx++)
        for (data = *mca++, bit = 0; bit < 8; bit++, data >>= 1)
            crc = (crc >> 1) ^ (((crc ^ data) & 1) ? crc32_POLY : 0);

    return crc & ((1 << HASH_BITS) - 1); /* return low bits for hash */
}
```

## C.2 Hash C Routine for Big Endian Architecture

```
#include <stdio>

unsigned HashIndex (char *Address);

main (int argc, char *argv[]) {

    int Index;
    char m[6];

    if (argc < 2) {
       printf("usage: hash xx-xx-xx-xx-xx-xx\n");
       return;
    }
```

```
    sscanf(argv[1],"%2X-%2X-%2X-%2X-%2X-%2X",
       &m[0],&m[1],&m[2],
       &m[3],&m[4],&m[5]);

    Index = HashIndex(&m[0]);

    printf("hash_index = %d  byte: %d   bit: %d\n",
            Index,Index/8,Index%8);
}


unsigned HashIndex (char *Address) {

     unsigned Crc = 0xffffffff;
     unsigned const POLY  0x04c11db6
     unsigned Msb;
     int BytesLength = 6;

     unsigned char CurrentByte;
     unsigned Index;
     int Bit;
     int Shift;

     for (BytesLength=0; BytesLength<6; BytesLength++) {

        CurrentByte = Address[BytesLength];

        for (Bit=0; Bit<8 ; Bit++) {

            Msb = Crc >> 31;
            Crc  <<= 1;

            if ( Msb ^ (CurrentByte & 1)) {

                 Crc ^= POLY;
                 Crc |= 0x00000001;
            }
            CurrentByte >>= 1;
        }
     }

    /* the hash index is given by the upper 9 bits of the CRC
     * taken in decreasing order of significance
     * index<0> = crc<31>
     * index<1> = crc<30>
     * ...
     * index<9> = crc<23>
     */
```

```
    for (Index=0, Bit=23, Shift=8;
        Shift >= 0;
        Bit++, Shift--)  {
           Index |= ( ( (Crc>>Bit) & 1 ) << Shift );
    }

    return Index;

}
```

# D

# Technical Support, Ordering, and Associated Literature

This appendix provides information about ordering Digital's microprocessor and peripheral chips, and associated literature. It also explains how to obtain technical support and information.

## D.1 Digital Semiconductor Information Line

If you need technical support or help deciding which literature best meets your needs, call the Digital Semiconductor Information Line:

| | |
|---|---|
| United States and Canada | **1–800–332–2717** |
| TTY (United States only) | **1–800–332–2515** |
| Outside North America | **+1–508–568–6868** |

## D.2 Ordering Digital Semiconductor Products

To order the following Digital Semiconductor products, contact your local distributor.

| Product | Order Number |
|---|---|
| DECchip 21041 PCI Ethernet LAN Controller | 21041–AA |
| DECchip 21041 PCI Evaluation Board Kit | 21A41–01 |

## D.3 Ordering Associated Literature

The following table lists DECchip 21041 literature that is available. For ordering information, contact the Digital Semiconductor Information Line.

| Title | Order Number |
|---|---|
| DECchip 21041 PCI Ethernet LAN Controller Product Brief | EC–QAWVA–TE |
| DECchip 21041 PCI Ethernet LAN Controller Data Sheet | EC–QAWWA–TE |
| Connecting the DECchip 21041 PCI Ethernet LAN Controller to the Network: An Application Note | EC–QJBWA–TE |
| DECchip 21040 Ethernet LAN Controller for PCI Product Brief | EC–N0281–72 |
| DECchip 21040 Ethernet LAN Controller for PCI Data Sheet | EC–N0280–72 |
| DECchip 21040 Ethernet LAN Controller for PCI Hardware Reference Manual | EC–N0752–72 |

## D.4 Ordering Third-Party Literature

You can order the following third-party literature directly from the vendor.

| Title | Vendor |
|---|---|
| PCI Local Bus Specification, Revision 2.0 | PCI Special Interest Group<br>M/S HF3–15A<br>5200 N.E. Elam Young Pkwy<br>Hillsboro, Oregon 97124–6497<br>(503) 696–2000 |
| Institute of Electrical and Electronics Engineers (IEEE) 802.3 | IEEE Service Center<br>445 Hoes Lane<br>P.O. Box 1331<br>Piscataway, NJ 08855–1331<br>1–800–678–IEEE (U.S. and Canada)<br>908–562–3805 (Outside U.S. and Canada) |

# Index

## H



## I



## J



## L



## M

## T



## W