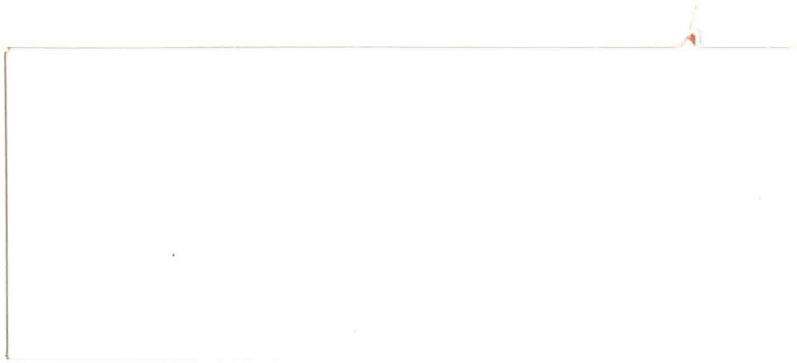
DECnet Digital Network Architecture Phase IV

Ethernet Data Link Functional Specification

Order No. AA-Y298A-TK

digital software



DECnet Digital Network Architecture Phase IV

Ethernet Data Link Functional Specification

Order No. AA-Y298A-TK

December 1983

This document describes the structure, functions, interfaces, and protocols of the DNA Ethernet Data Link not defined in the Ethernet Specification that make it compatible with DNA.

SUPERSESSION/UPDATE INFORMATION: This is a new manual.

VERSION: 1.0.0

To order additional copies of this document, contact your local Digital Equipment Corporation Sales Office.

digital equipment corporation · maynard, massachusetts

First Printing, December 1983

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

Distributed Systems Publications typeset this manual using DIGITAL's TMS-11 Text Management System.

MGTPEALL

CONTENTS

1 INTRODUCTION

The Digital, Intel, Xerox intercompany Ethernet specification is the basis for defining the DNA (DIGITAL Network Architecture) Ethernet Data Link. The DNA Ethernet Data Link incorporates the functions and operations defined in the Ethernet Specification Version 2.0. To these, the DNA Ethernet Data Link adds further features needed by the upper layers of DNA. This specification assumes understanding of the Ethernet specification.

This document describes the structure, functions, interfaces, and protocols of the DNA Ethernet Data Link not defined in the Ethernet specification that make it compatible with DNA. DNA is the model on which DECnet implementations are based. A DECnet network is a family of software modules, data bases, and hardware components used to tie DIGITAL systems together for resource sharing, distributed computation or remote system communication.

DNA is a layered structure. Modules in each layer perform distinct functions. Modules within a single DNA layer (but typically in different computer systems) communicate using specific protocols. Modules in different layers (but typically in the same computer system) interface using subroutine calls or a system-dependent method. In this document interfaces are described in terms of calls to subroutines.

This document assumes that the reader is familiar with computer communications and DECnet. The primary audience consists of those who implement DECnet systems. However, the document may be useful to anyone interested in the details of DECnet structure. The other current DNA functional specifications are:

DNA Data Access Protocol (DAP) Functional Specification, Version 5.6.0, Order No. AA-K177A-TK

DNA Digital Data Communications Message Protocol (DDCMP) Functional Specification, Version 4.1.0, Order No. AA-K175A-TK

DNA Ethernet Node Product Architecture Specification, Version 1.0.0, Order No. AA-X440A-TK

DNA Maintenance Operations Functional Specification, Version 3.0.0, Order No. AA-X436A-TK

DNA Network Management Functional Specification, Version 4.0.0, Order No. AA-X437A-TK

DNA Network Services Protocol Functional Specification, Version 4.0.0, Order No. AA-X439A-TK

DNA Routing Layer Functional Specification, Version 2.0.0, Order No. AA-X435A-TK

DNA Session Control Functional Specification, Version 1.0.0, Order No. AA-K182A-TK

The Ethernet; a Local Area Network; Data Link Layer and Physical Layer Specifications, Version 2.0, (Digital, Intel, and Xerox), Order No. AA-K759B-TK

The DECnet DIGITAL Network Architecture (Phase IV) General Description (Order No. AA-N149A-TC) provides an overview of the network architecture and an introduction to each of the DNA functional specifications.

2 FUNCTIONAL DESCRIPTION

The Ethernet functions are a subset of those included by the DNA Ethernet Data Link. The DNA Ethernet Data Link has two classes of functions:

- User -- the data communication services that DNA Ethernet Data Link provides for higher layers. In the context of this specification, the user is the next layer up in the network architecture, rather than the end user at the top layer.

- Network Management -- the control and observation services needed to maintain the Data Link.

The DNA Ethernet Data Link gives its users a communication service for transmitting and receiving frames, but does not guarantee delivery. The data link filters incoming frames by system address and protocol type based on filter values established by the user.

A specific system (physical address), a function-oriented group of systems (multicast address), or all systems (broadcast address) can be addressed using the data link. Each frame has a protocol type assigned to it that is used by the data link to identify the protocol handling module that is to receive it.

The use of protocol types allows concurrent operation of multiple protocols in the layer above the data link. For example, in DNA the Routing layer and the DNA Ethernet Data Link Loop Testing Service can both use the DNA Ethernet Data Link at the same time. Neither needs to be aware of the other. This is in contrast to the DDCMP Data Link where maintenance traffic and normal traffic use mutually exclusive modes of protocol operation.

The DNA Ethernet Data Link provides control and observation services needed to maintain the data link. These are functions that enable and disable physical channels and monitor the status of specific channels.

2.1 Design Scope

The DNA Ethernet Data Link requires certain characteristics to be present, attempts to meet certain goals, and lacks some features that

are not within the scope of the design.

2.1.1 Requirements

The DNA Ethernet Data Link design must have the following characteristics:

- All capabilities included in the Ethernet Specification are available from the DNA Ethernet Data Link.
- The network manager can control and observe the data link as it functions.
- It complies with the Ethernet Specification.

2.1.2 Goals

The DNA Ethernet Data Link design attempts to have the following characteristics:

- Uses both processor and memory efficiently.
- Common functions needed by higher layers are included in the data link.
- Inputs and outputs are simple, predictable, and consistent.

2.1.3 Non-Goals

DNA Ethernet Data Link design does not attempt to have the following characteristics:

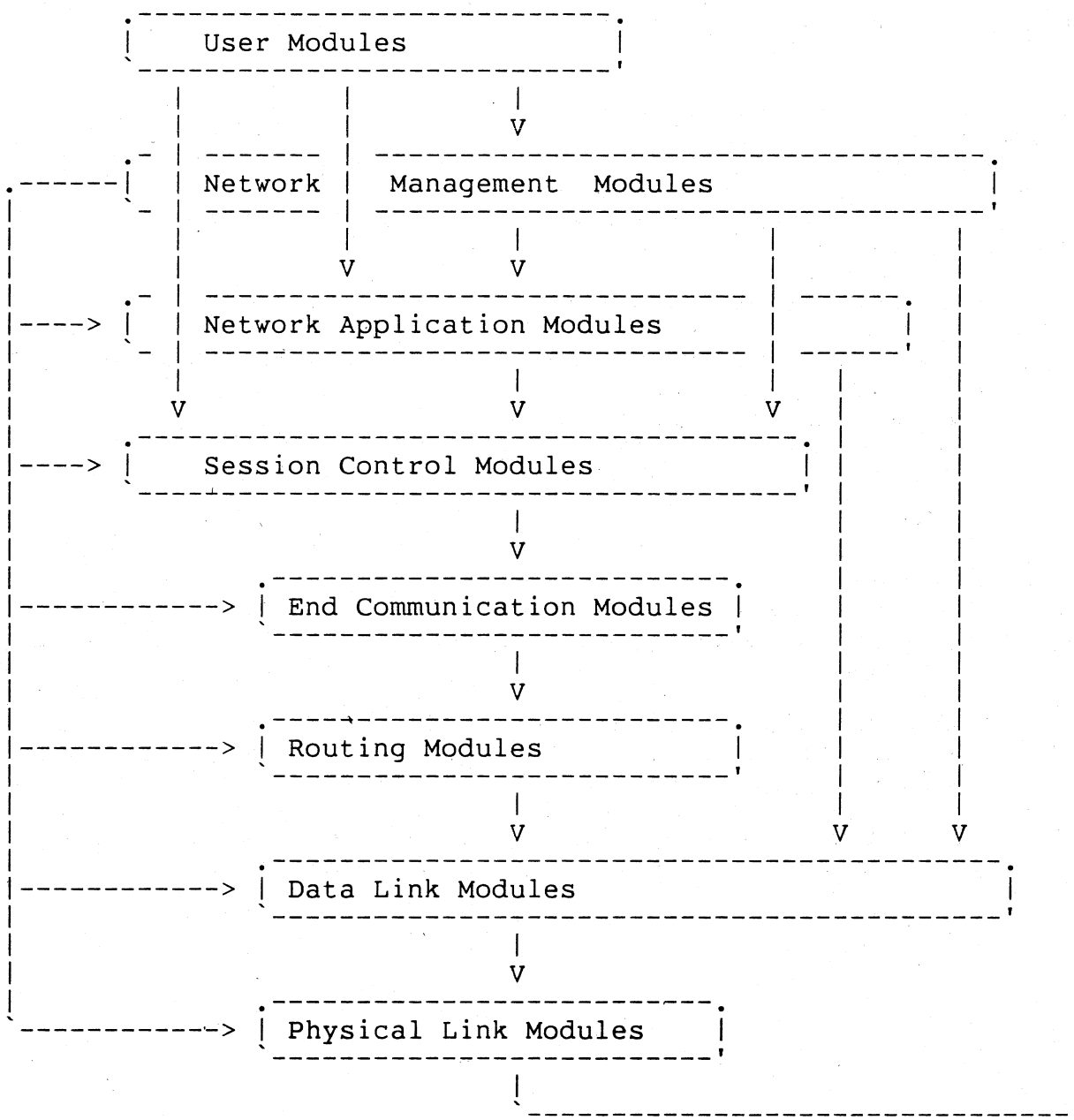
- Addition of functions not included in the Ethernet specification that higher layers may want to implement differently (such as reliable delivery).
- Self-diagnosis of all data link problems. Diagnosis of some problems requires information from higher layers.

3 MODELS

This section describes the relationship of the DNA Ethernet Data Link to other network layers and modules. Although this specification primarily relates the DNA Ethernet Data Link to DNA, the same relationships can also be applied within other network architectures.

3.1 Relationship to DIGITAL Network Architecture

Figure 1 shows the relationship of the DNA Ethernet Data Link to the DNA hierarchy.



The DNA Ethernet Data Link resides within the DNA Data Link layer. It can be co-resident with other DNA Data Link modules such as DDCMP or X.25.

In figure 1, horizontal arrows show direct access for control and observation of parameters, counters, etc. Vertical arrows show interfaces between layers for normal user operations such as file access, down-line load, and logical link usage.

Each layer in DNA consists of functional modules and protocols. Generally, modules use the services of the next lower layer. In this document the service relationship is demonstrated in the way the interfaces are modeled, as calls to subroutines. Note that the Network Management layer interfaces directly with each of the lower layers. Also, the layers above Session Control interface directly with it. For this reason the upper three layers are sometimes referred to as the "end user."

Modules of the same type in the same layer communicate with each other to provide their services. The rules governing this communication and the messages required constitute the protocol for those modules. Messages are typically exchanged between equivalent modules in different nodes. However, equivalent modules within a single node can also exchange messages.

A brief description of each layer follows in order from the highest to the lowest layer:

1. User layer. The highest layer, the User layer supports user services and programs. Programs such as the Network Control Program, which interfaces with the Network Management layer, and file transfer programs, which interface with the Network Application layer, reside in the user layer.

2. Network Management layer. The Network Management layer is the only one that has direct access to each lower layer for control purposes. Modules in this layer provide user control over and access to network parameters and counters. These modules also perform up-line dumping, down-line loading, and testing functions.

3. Network Application layer. Modules in the Network Application layer support network functions, such as remote file access and file transfer, used by the User and Network Management layers.

4. Session Control layer. The Session Control defines the system-dependent aspects of logical link communication, which allows messages to be sent from one node to another in a network. Session Control functions include name to address translation, process addressing, and, in some systems, process activation and access control.

5. End Communication layer. The End Communication layer defines the system-independent aspects of logical link communication.

6. Routing layer. Modules in the Routing layer route messages, called packets, between source and destination nodes.

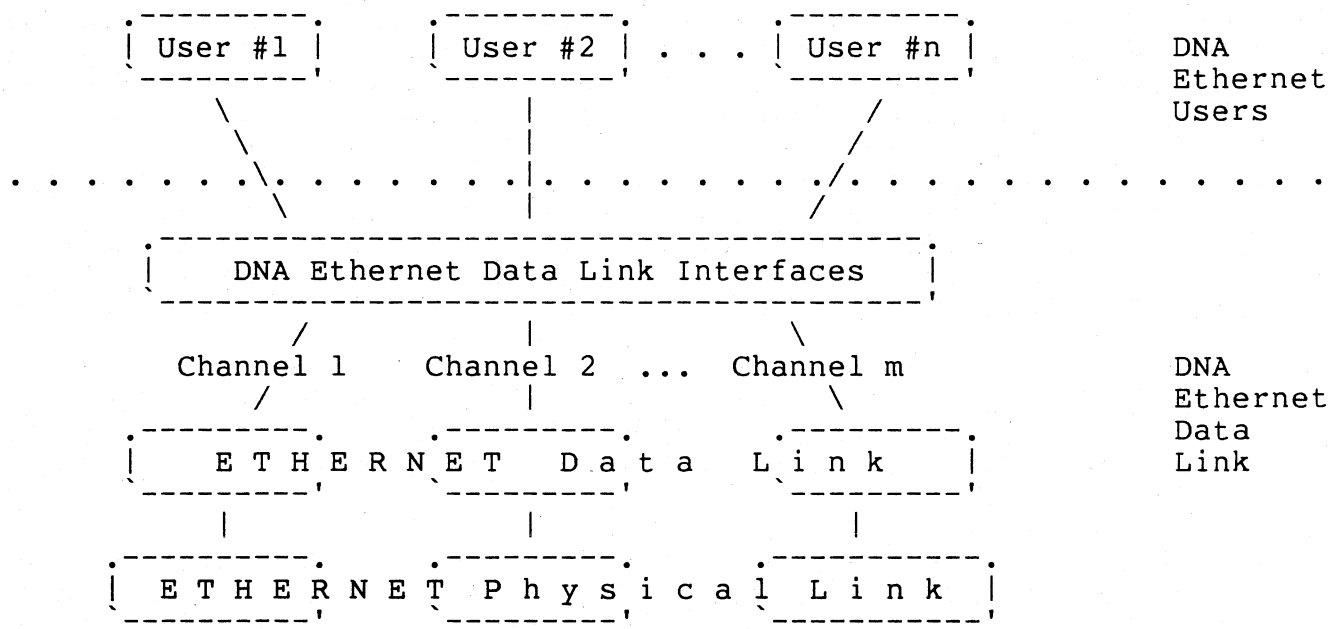
7. Data Link layer. The Data Link layer defines the protocol concerning data integrity and physical channel management.

8. Physical Link layer. The Physical Link layer encompasses a part of the device driver for each communications device plus the communications hardware itself. The hardware includes interface devices, modems, and the communication lines.

3.2 DNA Ethernet Data Link Model

The DNA Ethernet Data Link provides communication services for multiple concurrent users. It also supports multiple Ethernet channels. A channel is defined as a single hardware connection to an Ethernet cable. Each channel implements the cross-company Ethernet standard.

The following diagram shows the relationship of these users to the DNA Ethernet Data Link and its relationship to the standard Ethernet Data Link. In Ethernet terminology, the DNA Ethernet Data Link is the lowest level module in the Client Layer.



3.2.1 Resource Naming

The user of the DNA Ethernet Data Link is concerned with two different levels of identification. Users identify either a channel or a portal.

- A channel is a particular Ethernet hardware connection. Each channel has one name and each name identifies one channel.

- A portal represents a particular user's data base needed by the data link to service that user's requests. A portal is assigned by the data link in response to a user's initial call. The user then uses that portal in subsequent related calls. Many users can simultaneously have their own portal associated by the data link with the same channel.

  A portal data base contains the lists of protocol types and multicast addresses that the user has enabled for receipt of incoming frames. The user will receive frames only for enabled protocol types and multicast addresses. It also contains the lists of outstanding transmits and receives for the user.

3.2.2 Data Link States

In observing the operation of the data link, the user can see four channel states. Some of these states are reached due to network management commands, others due to data link operation. The states are:

- Off -- the channel is not available.

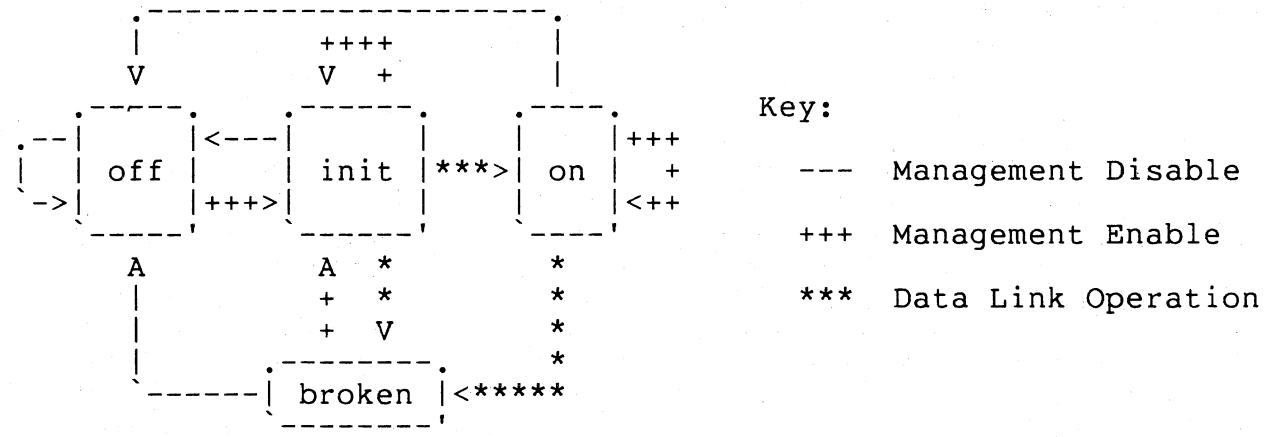
- Init -- the channel is being initialized and tested by the data link. This test is an implementation dependent self-test.

- On -- the channel is available for use.

- Broken -- an attempt was made to turn the channel on, but it failed the initialization test, or the channel was on but the data link determined that the channel would now fail the initialization test.

Changes between these states do not affect counters or parameters such as the node's physical address.

The following diagram shows the data link state transitions for a channel:



Network management can enable the channel. From the "off" or "broken" states, this enable causes a transition to "init". From the "on" state, enable has no effect.

In the "init" state, the data link performs initialization and test of the channel. If this succeeds, the data link changes the state to "on". If initialization and test fail, the data link changes the state to "broken".

From the "broken" state, network management can either try initialization again with an enable command or go to "off" with a disable command.

The data link can change the state from "on" to "broken" at any time if it determines that the channel would fail initialization and test.

The channel can be forced to "off" from any state by a network management disable command.

4 INTERFACES

The following sections describe the interfaces provided by DNA Ethernet Data Link. They are the following:

- User Interface -- contains the functions to transmit and receive data between data links.
- Network Management Interface -- contains the control and observation functions for the local data link.

The function descriptions are in terms of subroutines with input and output arguments. These subroutines are to be understood as abstract, functional descriptions. Actual implementations may vary, for example in synchronization techniques, as long as they provide the same functions. Furthermore, these are models for privileged, internal system interfaces. They are not necessarily to be used as high level user interfaces.

The interfaces use the two levels of identification defined in the Model section. Callers identify the object of a call in terms of either channel-id or portal-id.

In many of the interface functions, the caller can specify an Ethernet address. In some cases this address can be either a physical address or a multicast address. In other cases it is restricted to one or the other. In implementations that allow expression of one of the forms of address in a case where it is invalid, the implementation must reject the function request with an appropriate error return code.

The interface descriptions assume that buffers are passed in the form of a descriptor that contains buffer address, maximum buffer length, and, if applicable, length of information in buffer.

This section also contains an overview of the required interface functions from the intercompany Ethernet specification.

4.1 User Interface

This section describes the User Interface to the DNA Ethernet Data Link. The User Interface maintains portals for transmission and reception of frames through the data link on behalf of the user. The user can enable or disable broadcast and multicast addresses and appropriate protocol types on specific portals for flexible filtering of incoming frames. The User Interface provides functions for queuing frames and monitoring the status of those queued requests.

The interface contains the following functions:

- Open -- open a portal.
- Enable-promiscuous -- enable all protocol types and multicast addresses for a portal.
- Disable-promiscuous -- disable the blanket protocol type and multicast address enable for a portal.
- Enable-protocol -- enable a protocol type for a portal.
- Disable-protocol -- disable a protocol type for a portal.
- Enable-multicast -- enable a multicast address for a portal.
- Disable-multicast -- disable a multicast address for a portal.
- Close -- close a portal.
- Transmit -- send a frame.
- Transmit-poll -- checks for completion of a Transmit.
- Receive -- receive a frame.
- Receive-abort -- Aborts a Receive.
- Receive-poll -- checks for completion of a Receive.

Some features in the interface descriptions are optional. Features can be optional for caller or implementor or both. Caller-optional features can be invoked at the user's option. Implementor-optional features may or may not be included in particular implementations based on product requirements.

A portal will receive only those frames that match the filtering criteria set up with the enable and disable functions. These functions can be invoked multiple times for the same portal, allowing a single portal to receive many protocol types and multicast addresses. Alternatively, a portal can enable promiscuous receipt. This is exclusive of individual enables or disables, and allows the portal to receive all protocol types and multicast addresses.

Enabling of a protocol type automatically implies receipt of frames addressed to the channel's physical address. A portal receives multicast frames only for those multicast addresses it specifically enables. In this context, broadcast is treated the same as other multicast addresses. Multicast addresses enabled or disabled on one portal have no effect on other portals.

Conceptually, receive filtering is done first by protocol type, then by multicast address. A frame that does not pass filtering is discarded. In actual practice, an implementation may first filter in hardware the union of the multicast addresses for all portals and then

do the filtering again on a reduced number of frames.

4.1.1 Open

The open function opens a portal so that the user can transmit and receive frames.

OPEN(Channel-id,Pad,Return-code,Portal-id)

Inputs:

Channel-id - the unique identification of the Ethernet hardware on which the portal is to be opened.

Pad - a user flag that indicates whether the data link is to pad frames that are under minimum length. Padding is accomplished via data link generated additions to user data. It is the responsibility of the user's protocol conventions to define that its protocol modules in other systems will either request the same option or will understand the padding conventions. The data link will apply padding conventions to both transmitted and received messages for the portal.

Outputs:

Return-code - the status of the request. One of:

Success - a portal was opened.

No resources - the DNA Ethernet Data Link does not have sufficient resources to open a portal.

Unrecognized channel - there is no channel with the specified identification.

Channel not on - a portal cannot be opened because the channel state is not "on".

Promiscuous receiver active - a portal cannot be opened because some other portal has enabled promiscuous receive. This error return applies only in implementations that limit promiscuous receive to a single portal (see the Enable-promiscuous function).

Portal-id - a portal identification to be used in the other user interface functions.

4.1.2 Enable-promiscuous

The Enable-promiscuous function indicates that the portal is to receive all frames regardless of protocol type or multicast address. If a portal has this enable function in effect, it cannot explicitly enable individual protocol types or multicast addresses.

Three possible implementations of this function are allowed:

1. Not implemented. The function fails.

2. Exclusive use only. The function fails if any other portal is open or if this portal has any protocol types or multicast addresses enabled.

3. Non-exclusive use. The function always succeeds. Any frames that would have been delivered to another portal are duplicated and delivered to this one also. This is in addition to all other frames.

ENABLE-PROMISCUOUS(Portal-id,Return-code)

Inputs:

Portal-id - a portal identification assigned by the Open function.

Outputs:

Return-code - the status of the request. One of:

Success - promiscuous receive enabled.

Not implemented - the implementation does not support the requested function.

Non-exclusive - the implementation allows only exclusive promiscuous receipt and this portal or some other portal has a protocol type or multicast address enabled.

Unrecognized portal - there is no open portal with the specified identification.

Channel not on - the function failed because the channel state is not "on".

4.1.3 Disable-promiscuous

The Disable-promiscuous function indicates that the portal is no longer to receive frames for all protocol types and multicast addresses.

DISABLE-PROMISCUOUS(Portal-id,Protocol-type,Return-code)

Inputs:

Portal-id - a portal identification assigned by the Open function.

Protocol-type - a protocol type that is to be recognized for this portal.

Outputs:

Return-code - the status of the request. One of:

Success - promiscuous receive is not enabled for the portal.

Unrecognized portal - there is no open portal with the specified identification.

4.1.4 Enable-protocol

The Enable-protocol function adds a protocol type to the list of those that the portal wishes to receive. The function fails if the protocol type is enabled by some other portal.

ENABLE-PROTOCOL(Portal-id,Protocol-type,Return-code)

Inputs:

Portal-id - a portal identification assigned by the Open function.

Protocol-type - a protocol type that is to be recognized for this portal.

Outputs:

Return-code - the status of the request. One of:

Success - the protocol type is enabled.

Protocol type in use - this portal cannot enable the protocol type because some other portal already has it enabled.

No resources - the DNA Ethernet Data Link does not have sufficient resources to enable another protocol type.

Unrecognized portal - there is no open portal with the specified identification.

Promiscuous receive active - the function failed because this portal has enabled promiscuous receive. This error return applies only in implementations that limit promiscuous receive to a single portal (see the Enable-promiscuous function).

Channel not on - the function failed because the channel state is not "on".

4.1.5 Disable-protocol

The Disable-protocol function indicates that the portal is no longer to receive frames for a protocol type.

DISABLE-PROTOCOL(Portal-id,Protocol-type,Return-code)

Inputs:

Portal-id - a portal identification assigned by the Open function.

Protocol-type - a protocol type that is no longer to be recognized for this portal.

Outputs:

Return-code - the status of the request. One of:

Success - the protocol type is not enabled for the portal.

Unrecognized portal - there is no open portal with the specified identification.

4.1.6 Enable-multicast

The Enable-multicast function adds a multicast address to the list of those that the portal wishes to receive.

ENABLE-MULTICAST(Portal-id,Multicast-address,Return-code)

Inputs:

Portal-id - a portal identification assigned by the Open function.

Multicast-address - a multicast address that is to be recognized for this portal.

Outputs:

Return-code - the status of the request. One of:

Success - the multicast address is enabled.

No resources - the DNA Ethernet Data Link does not have sufficient resources to enable another multicast address for this portal.

Unrecognized portal - there is no open portal with the specified identification.

Promiscuous receive active - the function failed because this portal has enabled promiscuous receive. This error return applies only in implementations that limit promiscuous receive to a single portal (see the Enable-promiscuous function).

Channel not on - the function failed because the channel state is not "on".

4.1.7 Disable-multicast

The Disable-multicast indicates that the portal is no longer to receive frames for a multicast address.

DISABLE-MULTICAST(Portal-id,Multicast-address,Return-code)

Inputs:

Portal-id - a portal identification assigned by the Open function.

Multicast-address - a multicast address that is no longer to be recognized for this portal.

Outputs:

Return-code - the status of the request. One of:

Success - the multicast address is not enabled for the portal.

Unrecognized portal - there is no open portal with the specified identification.

4.1.8 Close

The Close function closes an open portal and releases all its resources. A portal cannot be closed unless all outstanding transmit or receive requests are completed.

CLOSE(Portal-id,Return-code)

Inputs:

Portal-id - a portal identification assigned by the Open function.

Outputs:

Return-code - the status of the request. One of:

Success - the portal is closed.

Unrecognized portal - there is no open portal with the specified identification.

Calls outstanding - there are uncompleted transmit or receive requests outstanding on the portal.

4.1.9 Transmit

The Transmit function queues a frame to be transmitted. The user tests for completion by using Transmit-poll. Transmission of a frame always succeeds or fails within such a small amount of time that an abort function is not necessary.

The user can have multiple outstanding Transmits, up to the limit allowed by the implementation.

NOTE

Ethernet does not guarantee attempted delivery of fewer than 46 bytes or more than 1500 bytes of user data. The DNA Ethernet Data Link may attempt transmission, but the result is unspecified.

TRANSMIT(Portal-id,Destination-address,Protocol-type,Input-buffer, CRC-32,Return-code)

Inputs:

Portal-id - a portal identification assigned by the Open function.

Destination-address - the address of the frame destination. This can be either a physical address or a multicast address. As a matter of good citizenship, users should

not transmit to the broadcast address unless the message is intended for all systems, regardless of their function or manufacturer.

Protocol-type - identifies the protocol at the receiving system.

Input-buffer - a buffer containing the data to be sent. If the data is shorter than the minimum Ethernet message size and padding is not enabled, or the data is longer than the maximum Ethernet message size, then the frame may not arrive at the destination. If padding is enabled there is no minimum size, and the maximum size is the Ethernet limit minus two bytes. Until the request is completed (as sensed via Transmit-poll), the user must not disturb the contents of the buffer.

CRC-32 - a caller- and implementation-optional 32-bit cyclic redundancy code that is to be used for this frame. This overrides the one normally supplied by the data link.

Outputs:

Return-code - the status of the request. One of:

Request accepted - DNA Ethernet Data Link will attempt to transmit the frame. Notification of completion is via the Transmit-poll function.

CRC control not available - the user attempted to supply a CRC and the implemention either does not support or allow the function.

No resources - the DNA Ethernet Data Link does not have sufficient resources to queue another transmit for this portal.

Unrecognized portal - there is no open portal with the specified identification.

Channel not on - the function failed because the channel state is not "on".

4.1.10 Transmit-poll

The Transmit-poll function checks for the completion of a transmit request. The data link transmits frames in the order in which the user submits them.

Successful completion of this function implies only that the local transmitter believes that it sent the frame. It has no implication relative to reception by the destination.

TRANSMIT-POLL(Portal-id,Return-code,Input-buffer,Error-detail,
Fault-distance)

Inputs:

Portal-id - a portal identification assigned by the Open function.

Outputs:

Return-code - the transmit request for this portal. One of:

Not complete - no outstanding transmit for this portal is done.

None outstanding - there are no outstanding transmits for this portal.

Transmit successful - a frame successfully left the local transmitter.

Transmit failed - the local transmitter could not transmit the frame.

Unrecognized portal - there is no open portal with the specified identification.

Channel left "on" state - the function failed because the channel is no longer in the "on" state.

Input-buffer - the buffer that was supplied in the Transmit function.

Error-detail - the caller-optional reason for a return-code of "transmit failed". Detailed explanation of these reasons is in the section on events. One of:

Excessive collisions
Carrier check failed
Short circuit
Open circuit
Frame too long
Remote failure to defer

Fault-distance - the caller-optional distance in bit times to a short or open circuit. This is meaningful only when error-detail is "short circuit" or "open circuit".

4.1.11 Receive

The Receive function queues a buffer to receive a frame.

RECEIVE(Portal-id,Receive-bad,Output-buffer,Return-code,Frames-lost)

Inputs:

Portal-id - a portal identification assigned by the Open function.

Receive-bad - a caller- and implementation-optional indication that frames are to be returned even though they contain invalid data. This includes frames with CRC or framing errors.

Output-buffer - a descriptor of a buffer to contain the received frame.

Outputs:

Return-code - the status of the request. One of:

Request accepted - If a message is received for the specified portal, the DNA Ethernet Data Link will put it into the buffer. Notification of completion is via the Receive-poll function.

Receive-bad not implemented - the call requested receipt of bad frames but the implementation does not support the function.

No resources - the DNA Ethernet Data Link does not have sufficient resources to queue another receive for this portal.

Unrecognized portal - there is no open portal with the specified identification.

Channel not on - the function failed because the channel state is not "on".

Frames-lost - the caller-optional count of the number of frames for this portal that were discarded because no buffer was available. Note that this count can be incorrect if frames were lost at a low enough level that portal filtering could not be done. This argument is not required in implementations where the buffering is always available.

4.1.12 Receive-poll

The Receive-poll function checks for the completion of a receive request. The data link gives received frames to the user in the order in which they arrived.

```
RECEIVE-POLL(Portal-id,Return-code,Error-detail,Destination-address,
             Source-address,Protocol-type,Output-buffer,
             Bytes-lost,CRC-32)
```

Inputs:

Portal-id - a portal identification assigned by the Open function.

Outputs:

Return-code - the status of the receive request. One of:

Not complete - no outstanding receive for this portal is done.

None outstanding - there are no outstanding receives for this portal.

Receive successful - a frame was successfully received into the buffer.

Receive with overrun - a frame was successfully received, but had to be truncated to fit into the buffer.

Length error - the received frame length was inconsistent with the length recorded by the padding conventions. The buffer contains the data received, including the padding information and truncated if it was too long to all fit in the buffer.

Invalid data - a frame was received with bad data. This return-code value is only seen if on the Receive function the caller requested delivery of bad frames.

Receive aborted - the user cancelled the receive request with the Receive-abort function.

Unrecognized portal - there is no open portal with the specified identification.

Channel left .bb "on" state - the function failed because the channel is no longer in the "on" state.

Error-detail - the caller-optional reason for a return-code of "invalid data". Detailed explanation of these reasons is in the section on events. One of:

Block check error
Framing error
Frame too long

Destination-address - the address to which the received frame was addressed.

Source-address - the address of the system that transmitted the received frame.

Protocol-type - the protocol type from the received frame.

Output-buffer - the received data.

Bytes-lost - the number of bytes lost when the return-code is "receive with overrun". This argument is optional both for user and implementor. If it is applicable but not implemented, it returns a value of "not implemented".

CRC-32 - the caller-optional 32-bit cyclic redundancy code that came into the data link with the frame.

4.1.13 Receive-abort

The Receive-abort function aborts all incomplete receive requests for the portal. The buffers are returned via the Receive-poll function. They may be returned as aborted or as normally completed.

RECEIVE-ABORT(Portal-id,Return-code)

Inputs:

Portal-id - a portal identification assigned by the Open function.

Outputs:

Return-code - the status of the request. One of:

Success - the request is now complete.

Unrecognized portal - there is no open portal with the specified identification.

None outstanding - there are no outstanding receives for this portal.

4.2 Network Management Interface

This section describes the interface used to control and observe operation of the DNA Ethernet Data Link. The Network Management Interface enables and disables channels and monitors the events and counters that provide information on network operation.

The Network Management Interface contains the following functions:

- Read-channel -- read channel status.
- Read-portal-list -- read list of open portals.

- Read-portal -- read information for a portal.
- Reset -- reset the channel to initial state.
- Set-address -- set channel physical address.
- Enable-channel -- enable channel operation.
- Disable-channel -- disable channel operation.
- Read-counters -- read channel or portal counters.
- Read-event -- read channel event.

4.2.1 Read-channel

The Read-channel function reads status information relative to a specified channel.

```
READ-CHANNEL(Channel-id,Return-code,Physical-address,
             Hardware-address,State,Broken-code)
```

Inputs:

Channel-id - the unique identification of a channel for which status is to be read.

Outputs:

Return-code - the status of the request. One of:

Success - status successfully read.

Unrecognized channel - there is no channel with the specified identification.

Physical-address - the physical address of the channel or "not set". This is the address that the channel is currently using.

Hardware-address - the hardware address of the channel or "not available". This is the address associated with the channel hardware.

State - the channel state as described earlier, one of:

On
Off
Init
Broken

Broken-code - an implementation-specific value indicating the

reason the state is "broken".

4.2.2 Read-portal-list

The Read-portal-list function reads the list of open portal identifications.

READ-PORTAL-LIST(Channel-id,Buffer,Return-code)

Inputs:

Channel-id - the unique identification of a channel for which the open portal list is to be read.

Buffer - a buffer to contain the list of portals.

Outputs:

Return-code - the status of the request. One of:

Success - list successfully read.

Buffer too small - the buffer could not hold the entire list. Portal identifications that would not fit in the buffer are not returned.

Unrecognized channel - there is no channel with the specified identification.

Buffer - descriptor of buffer containing list of portal identifications.

4.2.3 Read-portal

The Read-portal function reads portal data base information for a portal.

READ-PORTAL(Portal-id,Buffer,Return-code)

Inputs:

Portal-id - the portal identification of the open portal for which the data base is to be read.

Buffer - descriptor of a buffer to contain the portal data base information.

Outputs:

Return-code - the status of the request. One of:

Success - portal data base successfully read.

Buffer too small - the buffer could not hold all the information. Information that would not fit in the buffer is not returned.

Unrecognized channel - there is no channel with the specified identification.

Buffer - descriptor of a buffer containing the portal data base information. Each parameter entry contains the following information:

Parameter-type - the identification of the particular portal data base parameter. The parameters are listed in the Portal Data Base operation section.

Value - the parameter value.

4.2.4 Reset

The Reset function returns the specified channel to initial state. The physical address is unset. The counters are zeroed. All portals are closed. The channel state is "off".

RESET(Channel-id,Return-code)

Inputs:

Channel-id - the unique identification of the channel that is to be reset.

Outputs:

Return-code - the status of the request. One of:

Success - channel reset.

Unrecognized channel - there is no channel with the specified identification.

4.2.5 Set-address

The Set-address function sets the physical address of the specified channel. This is the address recognized as specific destination in

received frames and sent as source address in transmitted frames.

This function is necessary for a system that wants to use the same physical address on multiple different cables or has a physical address that is obtained completely independently of the data link.

SET-ADDRESS(Channel-id,Address,Return-code)

Inputs:

Channel-id - the unique identification of the channel for which the address is to be set. This must be a physical address. Furthermore, it must be unique among all channels attached to the same cable.

Address - the address to set.

Outputs:

Return-code - the status of the request. One of:

Success - the address was successfully set.

Unrecognized channel - there is no channel with the specified identification.

Channel not off - the function failed because the channel state is not "off".

4.2.6 Enable-channel

The Enable-channel function attempts to put the channel into a state where it can be used.

Enabling the channel does not disturb the contents of counter variables.

ENABLE-CHANNEL(Channel-id,Return-code)

Inputs:

Channel-id - the unique identification of the channel to be enabled.

Outputs:

Return-code - the status of the request. One of:

Success - Channel is enabled. A channel that is enabled is not necessarily usable. The user can determine the actual state of the channel with the

Read-channel function.

Address not set - the physical address for the channel has not been set.

Unrecognized channel - there is no channel with the specified identification.

4.2.7 Disable-channel

The Disable-channel function puts the channel into the "off" state so that it cannot be used. The purpose of this function is to halt operation. The channel can still be observed. In particular counters and other data base information not directly related to state change operation is not disturbed.

All outstanding transmit and receive operations for the channel are completed with a return code of "channel left on state" on the Receive-poll or Transmit-poll.

DISABLE-CHANNEL(Channel-id,Return-code)

Inputs:

Channel-id - the unique identification of the channel to be disabled.

Outputs:

Return-code - the status of the request. One of:

Success - channel is disabled.

Unrecognized channel - there is no channel with the specified identification.

4.2.8 Read-counters

The Read-counters function reads and optionally sets all counters associated with the specified channel or portal to zero.

READ-COUNTERS(Entity-id,Operation,Buffer,Return-code)

Inputs:

Entity-id - the unique identification of the particular channel or portal for which counters are to be read.

Operation - the data link performs one of the following

operations:

Read - only read the counters.

Read-and-zero - read the counters and set them to zero.

Buffer - descriptor of a buffer to receive the counters.

Outputs:

Return-code - the status of the request. One of:

Success - counters read (and set to zero if requested).

Unrecognized entity - there is no channel or portal with the specified identification.

Buffer too small - the buffer was too small to hold all the counter information. The information is truncated. Information that would not fit is lost. If read-and-zero was requested, all the counters are set to zero even if their values were not returned.

Buffer - descriptor of buffer containing counters. Each counter entry contains the following information:

Counter-type - the identification of the particular counter. The counters are listed in the Network Management Information section.

Value - the counter value.

Causes - specific reasons the counter was incremented. Some counters may be incremented for one or more reasons. For example, the data errors inbound counter can be incremented because of block check errors or framing errors.

4.2.9 Read-event

The Read-event function reads the next event from the event buffer associated with the specified channel and removes that event from the queue.

Only one event at a time need be buffered. The higher level must poll the event buffer often enough to avoid an excessive number of lost events.

READ-EVENT(Channel-id,Buffer,Return-code,Events-lost)

Inputs:

Channel-id - the unique identification of the channel for which an event is to be read.

Buffer - descriptor of a buffer to receive the event.

Outputs:

Return-code - the status of the request. One of:

Success - event read with no problems.

Buffer too small - the buffer was not big enough to contain the entire event. The information is truncated. Information that would not fit is lost.

No events - the queue is empty.

Unrecognized channel - there is no channel with the specified identification.

Events-lost - the number of events that were lost since the last Read-event.

Buffer - descriptor of buffer containing event. The event includes the following information:

Event-type - the identification of the event. The events and their parameters are listed in the Network Management Information section.

Event-parameters - the parameters associated with the event. Each event parameter entry includes the following information:

Parameter-type - the identification of the parameter.

Value - the value of the parameter.

4.3 Ethernet Interfaces

This section summarizes the Ethernet interfaces on which DNA Ethernet Data Link depends. For a complete description of the Ethernet interfaces, refer to the Digital, Intel, Xerox Ethernet Specification, Version 2.0. This section only contains those functions that relate directly to the operation of the DNA Ethernet Data Link.

The Ethernet specification uses Pascal as its representation language. That notation is reproduced here to the minimum extent necessary to recognize functions between the specifications. This is not intended to be a complete description of the Ethernet interfaces.

Most of the Ethernet interface functions return a status as the value of the function.

4.3.1 Client to Data Link Interface

The Client to Data Link Interface is the one used by the DNA Ethernet Data Link to transmit and receive frames. It contains the following functions:

- TransmitFrame - send a frame.
- ReceiveFrame - receive a frame.

4.3.1.1 TransmitFrame

TransmitFrame sends a frame. It does not return until the transmit either succeeds or fails.

TransmitFrame (destinationParam,sourceParam,typeParam,dataParam)

Inputs:

destinationParam - address of the destination system.

sourceParam - physical address of the local system.

typeParam - protocol type for the frame.

dataParam - data to go in the frame.

Outputs:

TransmitFrame - the status of the operation. One of:

transmitOK - frame successfully transmitted.

excessiveCollisionError - transmission failed.

4.3.1.2 ReceiveFrame

ReceiveFrame receives a frame. It does not return until a frame is received.

ReceiveFrame (destinationParam,sourceParam,typeParam,dataParam)

Inputs: none.

Outputs:

ReceiveFrame - the status of the operation. One of:

receiveOK - frame successfully received.

frameCheckError - frame block check failed.

alignmentError - frame did not contain an integral number of bytes.

destinationParam - address of the destination system.

sourceParam - physical address of the remote system.

typeParam - protocol type from the frame.

dataParam - data from the frame.

4.3.2 Network Management to Data Link Interface

The Network Management to Data Link Interface is the one used by DNA Ethernet Data Link for network management information and control access to the DNA Ethernet Data Link. It contains the following functions:

- ReadEthernetAddress - read the physical address.
- ReadNumberCollisions - read the number of collisions for the last TransmitFrame.
- DataLinkOn - start data link operation.
- DataLinkOff - stop data link operation.
- SetAddressMode - changes addressing mode between normal and promiscuous.
- MulticastOn - enable reception of multicast addresses.
- MulticastOff - disable reception of multicast addresses.

4.3.3 Network Management to Physical Link Interface

The Network Management to Physical Link Interface is the one used by the DNA Ethernet Data Link for network management information and control access to the Ethernet Physical Link. It contains the following function:

. CheckTransmit - checks status of last TransmitFrame.

5 NETWORK MANAGEMENT INFORMATION

This section describes the counters and events that the DNA Ethernet Data Link provides for the Network Management layer. Each description contains a conceptual definition of the information and a statement of its use. Precise operational definitions are in the Operation section.

5.1 Counters

The following counters are kept for each Ethernet channel. Some of them are also kept for portals. Counters are unsigned integers. All counters remain at their maximum value to indicate overflow.

Unless otherwise stated, all counters include both normal and multicast traffic. Furthermore, they include information for all protocol types. Frames received and bytes received counters do not include frames received with errors.

The following table contains the names and bit lengths of all the counters:

Length	Name
16	Seconds since last zeroed
32	Bytes received
32	Bytes sent
32	Frames received
32	Frames sent
32	Multicast bytes received
32	Multicast frames received
32	Frames sent, initially deferred
32	Frames sent, single collision
32	Frames sent, multiple collisions
16	Send failure *
16	Collision detect check failure
16	Receive failure *
16	Unrecognized frame destination
16	Data overrun
16	System buffer unavailable
16	User buffer unavailable

* Counter has multiple causes

The following counters are kept for each portal:

Seconds since last zeroed
Bytes received
Bytes sent
Frames received

Frames sent
User buffer unavailable

. Seconds since last zeroed

The number of seconds since the counters were last zeroed. The length is 16 bits.

Provides a frame of reference for the other counters by indicating the amount of time they cover.

. Bytes received

The total number of user data bytes successfully received. This does not include Ethernet data link headers. This number is the number of bytes in the Ethernet data field, which includes any padding or length fields when they are enabled. The length is 32 bits. These are bytes from frames that passed hardware filtering.

When the number of frames received is used to calculate protocol overhead, the overhead plus bytes received provides a measurement of the amount of Ethernet bandwidth (over time) consumed by frames addressed to the local system.

. Bytes sent

The total number of user data bytes successfully transmitted. This does not include Ethernet data link headers or data link generated retransmissions. This number is the number of bytes in the Ethernet data field, which includes any padding or length fields when they are enabled. The length is 32 bits.

When the number of frames sent is used to calculate protocol overhead, the overhead plus bytes sent provides a measurement of the amount of Ethernet bandwidth (over time) consumed by frames sent by the local system.

. Frames received

The total number of frames successfully received. The length is 32 bits. These are frames that passed hardware filtering.

Provides a gross measurement of incoming Ethernet usage by the local system. Provides information used to determine the ratio of the error counters to successful transmits.

. Frames sent

The total number of frames successfully transmitted. This does not include data link generated retransmissions. The length is 32 bits.

Provides a gross measurement of outgoing Ethernet usage by the local system. Provides information used to determine the ratio of the error counters to successful transmits.

. Multicast bytes received

The total number of multicast data bytes successfully received. This does not include Ethernet data link headers. This number is the number of bytes in the Ethernet data field. The length is 32 bits.

In conjunction with total bytes received, provides a measurement of the percentage of this system's receive bandwidth (over time) that was consumed by multicast frames addressed to the local system.

. Multicast frames received

The total number of multicast frames successfully received. The length is 32 bits.

In conjunction with total frames received, provides a gross percentage of the Ethernet usage for multicast frames addressed to this system.

. Frames sent, initially deferred

The total number of times that a frame transmission was deferred on its first transmission attempt. The length is 32 bits.

In conjunction with total frames sent, measures Ethernet contention with no collisions.

. Frames sent, single collision

The total number of times that a frame was successfully transmitted on the second attempt after a normal collision on the first attempt. The length is 32 bits.

In conjunction with total frames sent, measures Ethernet contention at a level where there are collisions but the backoff algorithm still operates efficiently.

. Frames sent, multiple collisions

The total number of times that a frame was successfully transmitted on the third or later attempt after normal collisions on previous attempts. The length is 32 bits.

In conjunction with total frames sent, measures Ethernet contention at a level where there are collisions and the backoff algorithm no longer operates efficiently.

NOTE

No single frame is counted in more than one of the above three counters.

- Send failures

  The total number of times a transmit attempt failed. The length is 16 bits. Each time the counter is incremented, a type of failure is recorded. When Read-counter function reads the counter, the list of failures is also read. When the counter is set to zero, the list of failures is cleared.

  In conjunction with total frames sent, provides a measure of significant transmit problems. All of the problems reflected in this counter are also captured as events.

  Following are the possible failures. More information on their meanings and use can be found in the section on events.

  Excessive collisions
  Carrier check failed
  Short circuit
  Open circuit
  Frame too long
  Remote failure to defer

- Collision detect check failure

  The approximate number of times that collision detect was not sensed after a transmission. The length is 16 bits.

  If this counter contains a number roughly equal to the number of frames sent, either the collision detect circuitry is not working correctly or the test signal is not implemented.

- Receive failures

  The total number of frames received with some data error. Includes only data frames that passed either physical or multicast address comparison. The length is 16 bits. This counter includes failure reasons in the same way as the send failure counter.

  In conjunction with total frames received, provides a measure of data related receive problems. All of the problems reflected in this counter are also captured as events.

  Following are the possible reasons. More information on their meaning and use can be found in the section on events.

  Block check error
  Framing error
  Frame too long

- Unrecognized frame destination

  The number of times a frame was discarded because there was no portal with the protocol type or multicast address enabled. This includes frames received for the physical address, the broadcast address, or a multicast address. The length is 16 bits.

- Data overrun

  The total number of times the hardware lost an incoming frame because it was unable to keep up with the data rate.

  In conjunction with total frames received, provides a measure of hardware resource failures. The problem reflected in this counter is also captured as an event.

- System buffer unavailable

  The total number of times no system buffer was available for an incoming frame. The length is 16 bits.

  In conjunction with total frames received, provides a measure of system buffer related receive problems. The problem reflected in this counter is also captured as an event.

  This can be any buffer between the hardware and the user buffers (those supplied on Receive requests). Further information as to potential different buffer pools is implementation specific.

- User buffer unavailable

  The total number of times no user buffer was available for an incoming frame that passed all filtering. These are the buffers supplied by users on Receive requests. The length is 16 bits.

  In conjunction with total frames received, provides a measure of user buffer related receive problems. The problem reflected in this counter is also captured as an event.

5.2 Events

The following events are recorded for each channel. Such information as channel identification, date, and time are assumed to be added as needed by higher layers.

The event descriptions include a conceptual definition of the event and an explanation of its use. Precise operational event definitions are in the Operation section.

Events are recorded regardless of protocol type.

NOTE

Since some events can occur very rapidly, implementations must take care that main line processing is not pre-empted by event processing.

. Initialization failed

This event is logged when the self test at initialization fails. The procedures in the self test are implementation dependent, but include such things as internal hardware loop testing. This event is optional on channels that cannot fail in initialization.

When this event is logged, it includes an implementation specific value that indicates the reason for the failure.

Provides notification that the channel is incapable of operating.

. Send failed

This event is logged for any error in transmitting a frame. The reason for the error is included, followed by any other pertinent data. The possible reasons are:

Excessive collisions

Exceeded the maximum number of retransmissions due to collisions.

Indicates an overload condition on the Ethernet. Too many systems are trying to transmit at the same time.

Carrier check failed

The data link did not sense the receive signal that is required to accompany the transmission of a frame.

Indicates a failure in either transmitting or receiving hardware. Could be either transceiver or transceiver cable related. Could be caused by a babbling controller that has been cut off.

Short circuit

There is a short somewhere in the local area network coaxial cable, or the transceiver or controller/transceiver cable failed. When this reason is logged, an estimated distance to the failure, in bit times, is included.

This indicates a problem either in the local hardware or a global network problem. The two can be distinguished by checking to see if other systems are reporting the same problem.

Open circuit

There is a break somewhere in the local area network coaxial cable. When this reason is logged, an estimated distance to the failure, in bit times, is included.

This indicates a problem either in the local hardware or a global network problem. The two can be distinguished by checking to see if other systems are reporting the same problem.

Frame too long

The controller or transceiver cut off transmission at the maximum size.

This indicates a problem with the local system: either it tried to send a frame that was too long, or the hardware cut off transmission too soon.

Remote failure to defer

A remote system began transmitting after the allowed window for collisions.

This indicates either a problem with some other system's carrier sense or a weak transmitter locally.

. Collision detect check failed

This event is logged when the data link did not sense the collision signal that is supposed to follow the transmission of a frame.

Indicates a failure either in the transceiver or transceiver cable collision circuitry.

. Receive failed

This event is logged for any error in receiving a frame. The reason for the error is included, followed by the source and destination physical addresses, the protocol type, and any other pertinent data as defined for the specific event, if available.

The possible reasons are:

Block check error

The frame failed the CRC check.

This indicates several possible failures. It can be caused by such things as electromagnetic interference, late collisions, or improperly set hardware parameters (such as receiver squelch).

Framing error

The frame did not contain an integral number of 8 bit bytes.

This indicates several possible failures. It can be caused by such things as electromagnetic interference, late collisions, or improperly set hardware parameters (such as receiver squelch).

Data overrun

The frame was lost due to a hardware resource failure.

This indicates, for example, insufficient hardware buffers or CPU time.

System buffer unavailable

The frame was discarded because there was no system buffer available to receive it.

This indicates a lack of buffers in the local system. This can be any buffer between the cable and the user buffers (those supplied by the user in the Receive function). Further information as to potential different buffer pools is implementation specific.

User buffer unavailable

The frame was discarded because there was no user buffer available to receive it.

This indicates a lack of buffers in the user process. These are the buffers supplied by the user in the Receive function.

Unrecognized frame destination

The frame was discarded because there was no portal with either the protocol type or the multicast address enabled. This includes frames received for the physical address, the broadcast address, or a multicast address.

This indicates either that the local system has not enabled a protocol type or multicast address that it should or that a remote system is attempting to use a protocol that is locally unsupported.

Frame too long

The frame was discarded because it was outside the Ethernet maximum length and could not be received.

This indicates that a remote system is sending invalid length frames.

6 INTERFACE USAGE EXAMPLES

This section contains examples of how the user interface might be used. The intent is to motivate the functions and to show how they might interrelate. This section does not specify how the interfaces must be used.

6.1 Portal Filter Setup

This example opens a portal and sets up its receive filter criteria. The portal wishes to receive frames of protocol types P1 and P2 that are addressed to the physical address and multicast address M1.

```
Open(Channel-id,Return-code,Portal-id)
IF Return-code = "success"
   Enable-protocol(Portal-id,P1,Return-code)
   IF Return-code = "success"
      Enable-protocol(Portal-id,P2,Return-code)
      IF Return-code = "success"
         Enable-multicast(Portal-id,M1,Return-code)
      ENDIF
   ENDIF
ENDIF
IF Return-code = "success"
   {Use the data link}
   Close(Portal-id,Return-code)
ELSE
   {Handle error}
ENDIF
```

6.2 Data Error Diagnostic

In this case, the data link is to be used by a diagnostic program that has a direct interest in the characteristics of incorrectly received messages. It wishes to supply its own cyclic redundancy code, find what code was received from the other end, and be able to analyze the bit patterns in damaged frames. Using these features it can test some of the data handling within both the local and the remote sides of the data link.

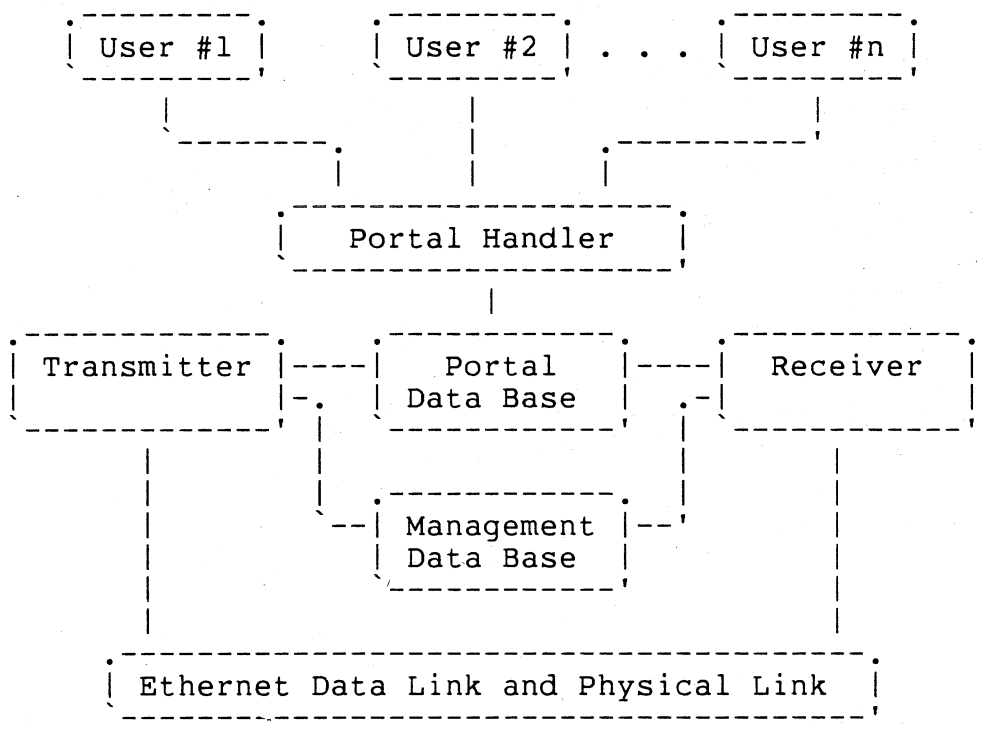
It accomplishes this by using the options on the Transmit, Receive, and Receive-poll functions that allow it to meet its task specific needs.

7 OPERATION

The bulk of the detailed operation of Ethernet is found in the Ethernet Specification, Version 2.0, and is not repeated here. This section specifies the operation of those functions that are additions to the Ethernet base. The operations specified here are portal handling and transmit/receive handling, which includes recording of counters and events.

This section uses a model implementation to describe DNA Ethernet Data Link operation. Implementations are not required to implement this model, but must be functionally equivalent. For example, a counter kept by one implementation must be incremented according to exactly the same criteria as the same counter in any other.

The model implementation is modularized according to the following diagram:



The Portal Handler maintains the Portal Data Base according to user requests through the User Interface. The Portal Handler communicates with the Transmitter and Receiver through the Portal Data Base. The Transmitter and Receiver send and receive frames through the DNA Ethernet Data Link User Interface and maintain counter and event information in the Management Data Base.

This specification assumes the mutual exclusion necessary to keep shared data bases correct.

The Portal Data Base contains an entry for each open portal. Each entry contains:

- Channel identification.
- Count of lost frames.
- Pad flag.
- List of enabled protocol types.
- List of enabled multicast addresses.
- List of outstanding Transmits, each list entry includes:
  - Request state, set to "pending" by the Portal Handler and "complete" by the Transmitter.
  - Destination address supplied by the user for the frame.
  - Protocol type supplied by the user for the frame.
  - Input buffer with the user's data to send in the frame.
  - Return code returned by the Transmitter.
  - CRC-32 optionally supplied by the user for the frame. If not user-supplied, the DNA Ethernet Data Link will supply it.
  - Error detail returned by the Transmitter if applicable and desired by the user.
  - Fault distance returned by the Transmitter if applicable and desired by the user.
- List of outstanding Receives, each list entry includes:
  - Request state, set to "pending" by the Portal Handler and "complete" by the Receiver.
  - Output buffer supplied by the user for a received frame.
  - Indicator whether user wants frames with invalid data.
  - Return code returned by the Receiver.
  - Error detail returned by the Receiver if applicable and desired by the user.
  - Destination address returned from the received frame by the Receiver.

- Source address returned from the received frame by the Receiver.
- Protocol type returned from the received frame by the Receiver.
- Bytes lost returned by the Receiver if applicable and desired by the user.
- CRC-32 returned from the received frame by the Receiver if desired by the user.

7.1 Portal Handler

This section describes Portal Handler operation relative to the User Interface functions. The Portal Handler functions are:

Open
Enable-promiscuous
Disable-promiscuous
Enable-protocol
Disable-protocol
Enable-multicast
Disable-multicast
Close
Transmit
Transmit-poll
Receive
Receive-abort
Receive-poll

7.1.1 Open

When the user calls Open, if the implementation allows only one promiscuous receiver, the Portal Handler checks to see if one is active. If so, it returns "promiscuous receiver active".

If there is no promiscuous receiver problem, the Portal Handler checks that it has the necessary resources to open a new portal. If not, it returns "no resources".

If resources are available, the Portal Handler checks to see if the requested channel exists. If not, it returns "unrecognized channel".

If the channel exists, the Portal Handler checks to see if it is on. If not, it returns "channel not on".

If the channel is on, the Portal Handler initializes the portal data base with the pad flag, empty lists, and the channel identification and returns "success" and the portal identification.

7.1.2 Enable-promiscuous

When the user calls Enable-promiscuous, the Portal Handler checks to see if the supplied portal identification identifies an open portal. If not, it returns "unrecognized portal".

If the portal is open, the Portal Handler does one of the following:

- Returns "not implemented".
- Or checks to see if any other portal is open or this one has any protocol types or multicast addresses enabled and if so it returns "non-exclusive".
- Or accepts the fact that for this portal it will have to duplicate messages that other portals receive.

If the Portal Handler can allow promiscuous receive, it enables promiscuous addressing through the DNA Ethernet Data Link. If this fails, it returns "channel in wrong state". Otherwise it puts "promiscuous" in the portal's list of enabled protocol types and returns "success".

7.1.3 Disable-promiscuous

When the user calls Disable-promiscuous, the Portal Handler checks to see if the supplied portal identification identifies an open portal. If not, it returns "unrecognized portal".

If the portal is open, the Portal Handler checks the portal's list of protocol types for "promiscuous". If not found, it returns "success".

If the portal is receiving promiscuously, the Portal Handler scans all other portals to see if any others are receiving promiscuously (if the implementation allows multiple promiscuous receivers). If none are found, the Portal Handler disables promiscuous addressing through the DNA Ethernet Data Link. The Portal Handler then returns "success".

7.1.4 Enable-protocol

When the user calls Enable-protocol, the Portal Handler checks to see if the supplied portal identification identifies an open portal. If not, it returns "unrecognized portal".

If the portal is open, the Portal Handler checks to see if it has resources to record another protocol type for this portal. If not, it returns "no resources".

If the Portal Handler has resources, it scans all portals to see if any of them have the protocol type enabled. If so, it returns "protocol type in use".

If the protocol type is available, the Portal Handler checks to see if the portal's channel is on. If not, it returns "channel not on".

If the channel is on, the Portal Handler puts the protocol type in the portal's list and returns "success".

7.1.5 Disable-protocol

When the user calls Disable-protocol, the Portal Handler checks to see if the supplied portal identification identifies an open portal. If not, it returns "unrecognized portal".

If the portal is open, the Portal Handler checks to see if the protocol type is in the portal's list. If it is, the Portal Handler removes it. The Portal Handler then returns "success".

7.1.6 Enable-multicast

When the user calls Enable-multicast, the Portal Handler checks to see if the supplied portal identification identifies an open portal. If not, it returns "unrecognized portal".

If the portal is open, the Portal Handler checks to see if it has resources to record another multicast address for this portal. If not, it returns "no resources".

If the Portal Handler has resources, it checks to see if the portal's channel is on. If not, it returns "channel not on".

If the channel is on, the Portal Handler puts the multicast address in the portal's list. If this is the first multicast address enabled, the Portal Handler calls the DNA Ethernet Data Link to enable multicast. The Portal Handler then returns "success".

7.1.7 Disable-multicast

When the user calls Disable-multicast, the Portal Handler checks to see if the supplied portal identification identifies an open portal. If not, it returns "unrecognized portal".

If the portal is open, the Portal Handler checks to see if the multicast address is in the portal's list. If it is, the Portal Handler removes it. If this was the only multicast address enabled, the Portal Handler calls the DNA Ethernet Data Link to disable multicast. The Portal Handler then returns "success".

7.1.8 Close

When the user calls Close, the Portal Handler checks to see if the supplied portal identification identifies an open portal. If not, it returns "unrecognized portal".

If the portal is open, the Portal Handler checks to see if the portal's transmit and receive lists are empty. If they are not, it returns "Calls outstanding".

If the portal's transmit and receive lists are empty, the Portal Handler releases all resources for the portal and returns "success".

7.1.9 Transmit

When the user calls Transmit, the Portal Handler checks to see if the supplied portal identification identifies an open portal. If not, it returns "unrecognized portal".

If the portal is open, the Portal Handler checks to see if it has resources to queue a transmit for the portal. If it does not, it returns "no resources".

If the Portal Handler has resources, it checks to see if the portal's channel is on. If it is not, it returns "channel not on".

If the portal's channel is on, the Portal Handler transfers the information from the call into the portal's transmit list, marked "pending", for action by the Transmitter. The Portal Handler then returns "request accepted".

7.1.10 Transmit-poll

When the user calls Transmit-poll, the Portal Handler checks to see if the supplied portal identification identifies an open portal. If not, it returns "unrecognized portal".

If the portal is open, the Portal Handler checks the transmit list to see if it is empty. If so, it returns "none outstanding".

If the transmit list is not empty, the Portal Handler checks to see if the first request state is "complete". If not, it returns "not complete".

If the first request is complete, the Portal Handler gets the output parameters from the list for return. The Portal handler then deallocates the transmit list entry and returns the output parameters.

7.1.11 Receive

When the user calls Receive, the Portal Handler checks to see if the supplied portal identification identifies an open portal. If not, it returns "unrecognized portal".

If the portal is open, the Portal Handler gets the portal's count of frames lost and zeros its own record of the count. The Portal Handler then checks to see if it has resources to queue a receive for the portal. If it does not, it returns "no resources" and the frames lost count.

If the Portal Handler has resources, it checks to see if the portal's channel is on. If it is not, it returns "channel not on" and the frames lost count.

If the portal's channel is on, the Portal Handler transfers the information from the call into the portal's receive list, marked "pending", for action by the Receiver. The Portal Handler then returns "request accepted" and the frames lost count.

7.1.12 Receive-abort

When the user calls Receive-abort, the Portal Handler checks to see if the supplied portal identification identifies an open portal. If not, it returns "unrecognized portal".

If the portal is open, the Portal Handler checks the receive list to see if it is empty. If so, it returns "none outstanding".

If the receive list is not empty, for each entry, the Portal Handler checks to see if the entry state is "complete". If not, it marks the request "complete", and sets the return code to "receive aborted". The Portal Handler then returns "success".

7.1.13 Receive-poll

When the user calls Receive-poll, the Portal Handler checks to see if the supplied portal identification identifies an open portal. If not, it returns "unrecognized portal".

If the portal is open, the Portal Handler checks the receive list to see if it is empty. If so, it returns "none outstanding".

If the receive list is not empty, the Portal Handler checks to see if the first request state is "complete". If not, it returns "not complete".

If the first request is complete, the Portal Handler gets the output parameters from the list for return. The Portal handler then deallocates the receive list entry and returns the output parameters.

7.2 Transmitter and Receiver

7.2.1 Transmitter

The Transmitter constantly scans the the transmit lists for all open portals for the first pending request. Whenever it finds one marked "pending" it uses the information from that entry to send the frame.

If the pad flag for the portal indicates that padding is enabled, the Transmitter prefixes the user's data with two bytes containing the length of the data as indicated by the user, low order byte first. If the resulting data is less than the Ethernet minimum, the Transmitter adds bytes of zeroes at the end of the user data to pad out to minimum length.

The Transmitter then uses the DNA Ethernet Data Link to send the frame. When this request completes, the transmitter records the necessary completion information in the portal's transmit list entry and marks it "complete". The Transmitter then continues its scan with the next portal.

7.2.2 Receiver

This operational description assumes that Receiver operation is fast enough to receive all frames received by the DNA Ethernet Data Link. Implementations that are not fast enough will have to be able to record frames lost due to no buffer.

This description also assumes the most complex form of protocol type matching. This is the form in which multiple users are allowed to receive promiscuously regardless of protocol types enabled by other users. The simpler cases are direct subsets of this case.

The Receiver has at least one Ethernet maximum size receive buffer of its own. The Receiver always tries to keep an Ethernet receive outstanding.

When a frame is received, the Receiver scans the open portals for a matching receiver. It checks the protocol type list first. If it finds a match here it may then check the multicast address list. If no match in the protocol type list, it goes on to the next portal. A receiver matches if one of the following is true:

- The protocol type matches and the frame destination is the physical address or the broadcast address.
- The protocol type matches and the multicast destination is in the portal's multicast address list.
- The portal protocol type list indicates promiscuous.

If the Receiver finds a match it checks the portal's receive list for the first pending request. If it does not find one it increments the portal's frames lost count and the total user buffer unavailable count, then proceeds as if it had completed giving the frame to the portal.

If the Receiver finds a pending receive, it copies the pertinent information into the portal's receive list according to the user's desires. For example it copies in a frame with invalid data (e.g. bad CRC) if the user requested this type of receive.

If the pad flag is set, the Receiver uses the first two bytes of the data as the received length and copies that amount into the user's buffer, not including the first two bytes. If the actual length of the data in the frame is less than the amount indicated by the first two bytes, the Receiver gives the user all of the data, including the first two bytes, and sets a "length error" return code.

The Receiver then marks the receive "complete".

The Receiver repeats this procedure until it has checked all open portals.

7.2.3 Counters

This section defines exactly when the data link increments counters. All counters operate the same with respect to overflow. Whenever a counter reaches maximum value for its length it remains at that value until the counters are set to zero. This operation is assumed in all of the following descriptions of when counters are incremented. All counters are cleared simultaneously.

NOTE

The names used in the Ethernet Version 2.0 specification are indicated in parentheses. A more formal description of their operation can be found in the Pascal procedural model found in that specification.

- Seconds since last zeroed (not specified in Ethernet V2.0)

  This counter is incremented once every second. It allows the counters to be maintained for about 18 hours without being read.

- Bytes received (not specified in Ethernet V2.0)

  This counter is updated every time a frame is received with no errors (framesReceivedNoErrors). It is incremented by the number of bytes in the Ethernet data field of the received frame (dataSize).

- Bytes sent (not specified in Ethernet V2.0)

  This counter is updated every time a frame is transmitted with no errors (frameSentNoErrors). It is incremented by the number of bytes in the Ethernet data field of the transmitted frame (dataSize).

- Frames received (framesReceivedNoErrors)

  This counter is incremented by one every time a frame is received with no errors.

- Frames sent (framesSentNoErrors)

  This counter is incremented by one every time a frame is transmitted with no errors.

- Multicast bytes received (not specified in Ethernet V2.0)

  This counter is updated every time a frame is received for a multicast address with no errors (framesReceivedNoError and address[1] = 1). It is incremented by the number of bytes in the Ethernet data field of the received frame (dataSize).

- Multicast frames received (not specified in Ethernet V2.0)

  This counter is incremented by one every time a frame is received for a multicast address with no errors (framesReceivedNoError and address[1] = 1).

- Frames sent, initially deferred (not specified in Ethernet V2.0)

This counter is incremented by one every time a frame is transmitted with no errors after being initially deferred. It does not get incremented if a deferral takes place on a subsequent attempt to transmit after a collision if the first attempt was not deferred.

. Frames sent, single collision (transmitOkOneCollision)

This counter is incremented by one every time a frame is transmitted with no errors after a single collision and backoff sequence; i.e., successful on the second attempt.

. Frames sent, multiple collisions (transmitOkMultipleCollisions)

This counter is incremented by one every time a frame is transmitted with no errors after more than one collision and backoff sequence; i.e., successful on the third or subsequent attempt.

. Send failures (not specified in Ethernet V2.0)

This counter is incremented by one every time an error causes the termination of the transmission of a frame. This may be due to one or more of the following faults occurring during a single Data Link TransmitFrame operation. A flag is set to indicate which type of fault(s) has occurred. (Ethernet V2.0 specifies two separate 16-bit counters for framesAbortedLateCollision and framesAbortedExcessCollisions.)

Excessive collisions (excessiveCollisionError)
Carrier check failed (carrierSenseFailed)
Short circuit
Open circuit
Frame too long
Remote failure to defer (lateCollision)

. Collision detect check failures (collisionDetectFailed)

This counter is incremented by one every time no collisionDetect signal is seen from the Physical Layer during a transmission or within 2 microseconds following the deassertion of carrierSense after the end of transmission.

. Receive failures (not specified in Ethernet V2.0)

This counter is incremented by one every time an error causes an incoming frame to be lost. This may be due to one or more of the following faults occurring during a single Data Link ReceiveFrame operation. A flag is set to indicate which type of fault(s) has occurred. (Ethernet V2.0 specifies two separate 16-bit counters for framesReceivedCRCErrors and framesReceivedAlignErrors)

Block check error (frameCheckError)
Framing error (alignmentError)
Frame too long

- Unrecognized frame destinations (not specified in Ethernet V2.0)

  This counter is incremented by one every time a frame is received but discarded because there was no portal with the protocol type or multicast address enabled.

- Data overruns (not specified in Ethernet V2.0)

  This counter is incremented by one every time an incoming frame is lost because the hardware was unable to keep up with the data rate.

- System buffers unavailable (not specified in Ethernet V2.0)

  This counter is incremented by one every time a frame is received but discarded because there is no system buffer available.

- User buffers unavailable (not specified in Ethernet V2.0)

  This counter is incremented by one every time a frame is received but discarded because there is no user buffer available.

7.2.4 Events

This section defines exactly when the data link records events.

NOTE

> The names used in the Ethernet Version 2.0 specification are indicated in parentheses. A more formal description of their operation can be found in the Pascal procedural model found in that specification.

- Initialization failed (not specified in Ethernet V2.0)

  This event is logged whenever the self test at initialization fails. The operation of the self test is implementation specific.

- Send failed (not specified in Ethernet V2.0)

  This event is logged every time an error causes the termination of the transmission of a frame. This may be due to one or more of the following faults occurring during a single Data Link TransmitFrame operation.

  Excessive collisions (excessiveCollisionError)
  Carrier check failed (carrierSenseFailed)
  Short circuit
  Open circuit

Frame too long
Remote failure to defer (lateCollision)

- Collision detect check failed (collisionDetectFailed)

  This event is logged every time no collisionDetect signal is seen from the Physical Layer during a transmission or within 2 microseconds following the deassertion of carrierSense after the end of transmission.

- Receive failed

  This event is logged every time an incoming frame is lost. This may be due to one or more of the following faults occurring during or immediately after a Data Link ReceiveFrame operation.

  Block check error (frameCheckError)
  Framing error (alignmentError)
  Data Overrun
  System buffer unavailable
  User buffer unavailable
  Unrecognized frame destination

APPENDIX A

PROTOCOL TYPES AND MULTICAST ADDRESSES

This appendix lists all the protocol types and multicast addresses defined for Digital Equipment Corporation or across companies. DIGITAL protocol types are in the range 60-00 through 60-09. DIGITAL physical and multicast addresses are in the range AA-00-00-00-00-00 through AA-00-04-FF-FF-FF and AB-00-00-00-00-00 through AB-00-04-FF-FF-FF, respectively.

A.1 CROSS-COMPANY ASSIGNMENTS

The cross-company protocol type is:

Value	Meaning
90-00	Loopback

The cross-company multicast addresses are:

Value	Meaning
FF-FF-FF-FF-FF-FF	Broadcast
CF-00-00-00-00-00	Loopback Assistance

A.2 DIGITAL ASSIGNMENTS

The DIGITAL protocol types are:

Value	Meaning
60-00	
60-01	DNA Dump/Load (MOP)
60-02	DNA Remote Console (MOP)
60-03	DNA Routing
60-04	Local Area Transport (LAT)
60-05	Diagnostics
60-06	Customer use
60-07	System Communication Architecture (SCA)

DNA Phase IV nodes have addresses in the range AA-00-04-00-00-00 through AA-00-04-00-FF-FF (see DNA Routing Layer Functional Specification).

The DIGITAL multicast addresses are:

Value	Meaning
AB-00-00-01-00-00	DNA Dump/Load Assistance (MOP)
AB-00-00-02-00-00	DNA Remote Console (MOP)
AB-00-00-03-00-00	DNA Routing Layer routers
AB-00-00-04-00-00	DNA Routing Layer end nodes
AB-00-03-00-00-00	Local Area Transport (LAT)
AB-00-04-00-00-00 thru AB-00-04-00-FF-FF	Customer use
AB-00-04-01-00-00 thru AB-00-04-01-FF-FF	System Communication Architecture (SCA)

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify)_____

Name_____ Date_____

Organization_____

Street_____

City_____ State_____ Zip Code_____
or
Country

—Do Not Tear - Fold Here and Tape —

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE DOCUMENTATION
1925 ANDOVER STREET TW/E07
TEWKSBURY, MASSACHUSETTS 01876

— Do Not Tear - Fold Here and Tape —

Cut Along Dotted Line

Printed in U.S.A.