ODI Specification Supplement: The Receive Monitor

Document Version 1.00
Part Number: 107-000057-001
24 November 1993

Disclaimer

Novell, Inc. makes no representations or warranties with respect to the contents or use of this manual, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Novell, Inc.
122 East 1700 South
Provo, Utah 84606

Trademarks

Novell has made every effort to supply trademark information about company names, products, and services mentioned in this document. Trademarks indicated below were derived from various sources.

Novell and NetWare are registered trademarks of Novell, Inc.

Internetwork Packet Exchange, ODI, Open Data-Link Interface, LSL, Link Support Layer, MLID, Multiple Link Interface Driver, MLI, Multiple Link Interface, MPI, Multiple Protocol Interface, MSM, Media Support Module, TSM, Topology Support Module, HSM, Hardware Support Module, RX-Net, NE1000, NE2000, NE/2, NE2-32, and NTR2000 are trademarks of Novell, Inc.

Table Of Contents

List of Figures

Overview

This chapter discusses the receive monitor. The receive monitor monitors all packets the MLID receives. The receive monitor can review all packets, all receive packets, or all transmit packets.

You should read this chapter if you are developing a protocol stack, such as a LANalyzer, that must review all packets on the medium.

Introduction to the Receive Monitor

The receive monitor examines every packet the MLID receives. It monitors the packets in one of two ways:

- It receives the look-ahead buffer, if the MLID supports the look-ahead reception method and the MLID is not going to pass the packet to the LSL.
- It receives every packet that the MLID receives, if the MLID does not use the look-ahead method. In this case, the receive monitor receives the whole packet, regardless of whether or not it was to be passed to the LSL and whether it is errored.

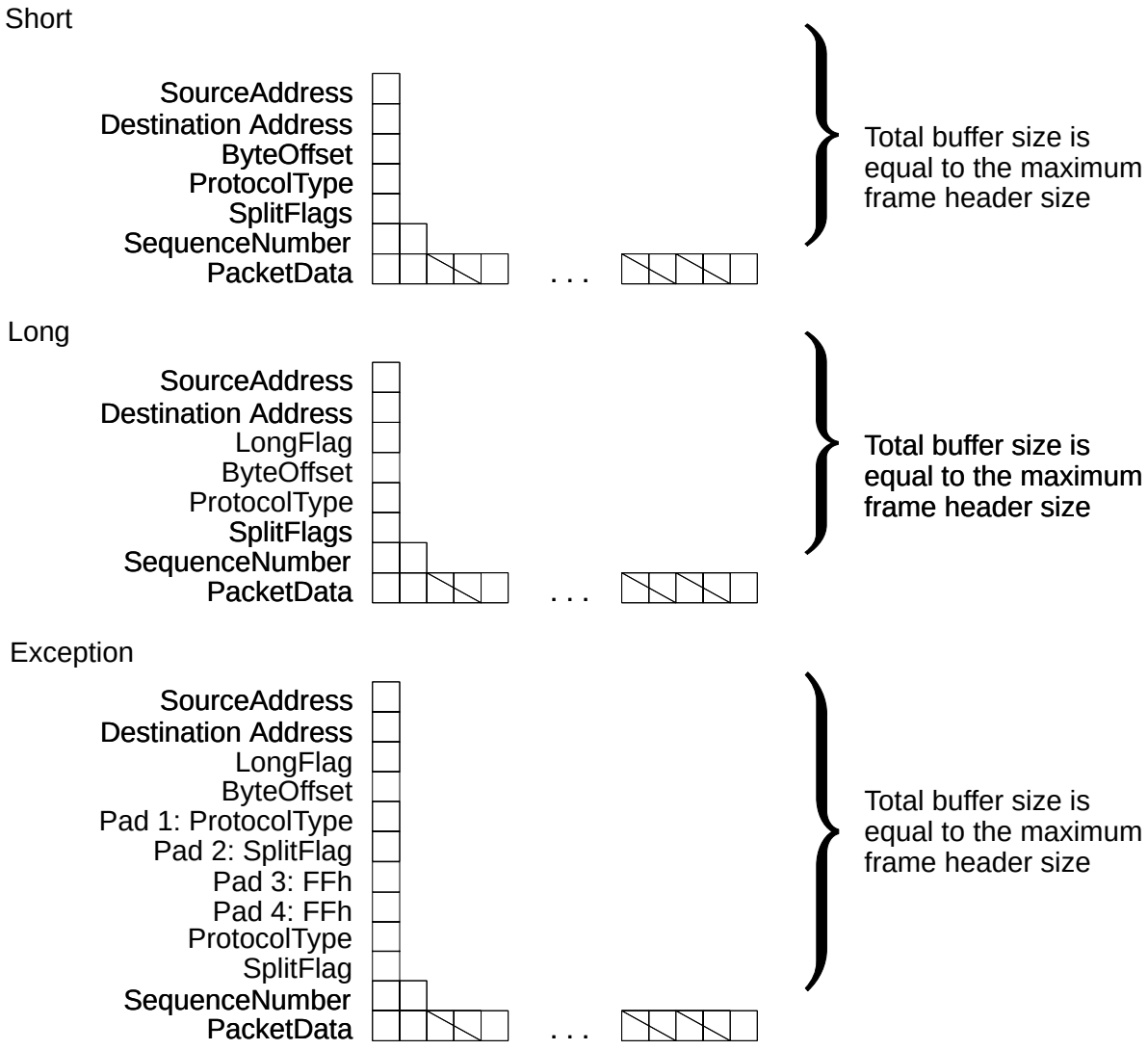
Look-ahead Packet Reception

When an MLID receives a packet, it fills out a *LookAheadStructure* that either points to the whole packet or the first "look-ahead" bytes of the packet, depending on whether the MLID is using the receive look-ahead packet reception method or is passing the entire packet to the LSL. The MLID then passes this *LookAheadStructure* to the receive monitor, if one is registered. If a receive monitor is not registered, the MLID handles the packet in the regular manner.

If the receive monitor must have more of the packet, it returns in ECX how much more it needs, and in EBX how many bytes to skip over. It also returns an ECB with a fragment structure pointing to the buffer that the MLID is to copy the rest of the packet into.

If the MLID is for RX-Net, the look-ahead buffer has the following format:

Figure 1
Format of RX-Net LookAhead Buffer



Packet Validity Rules

The receive monitor uses the following rules to determine the validity of the packets that are presented at the promiscuous mode interface.

Media Header Length Rules

The following rules govern the length of the various media headers.

- Token-Ring MAC packets that are not long enough to contain a major vector have media header lengths of 0.
- Token-Ring 802.2 packets that are not long enough to contain the 3 byte (u-frame) or 4 byte (i/s frame) 802.2 header have media header lengths of 0.

- Token-Ring 802.2 packets have media header lengths between 17 and 36 bytes, depending on the source route field length and the control field.
- Token-Ring SNAP packets that are not long enough to contain the 8 byte SNAP header have media header lengths of 0.
- Token-Ring SNAP packets have media header lengths between 22 and 40 bytes, depending on the source route field length.
- Ethernet packets with less than 22 bytes of data have media header lengths of 0.
- Ethernet SNAP frames have media header lengths of 22 bytes.
- Ethernet 802.2 frames have media header lengths of either 17 or 18 bytes, depending on the control field.
- Ethernet II frames have media header lengths of 14 bytes.
- Raw 802.3 frames have media header lengths of 14 bytes.

Packet Rules

Packets are created based upon specific variables for each topology. The following tables describe the rules that govern packet creation for each topology. The tables note whether the packet should be delivered to the protocol stack, the status bits in EAX that should be set for the packet, and the frame type designated by the rule (if any).

Note In the following tables, the ! symbol is a NOT symbol.

Ethernet Packets

Ethernet packets are created based upon the variants of packet size, length in 802.3 header, CRC error status, and packet type.

Table 1
Ethernet Packet Length Rules

Ethernet Packet Length Rules			
Condition	**Deliver to Stack**	**Status Bit Set**	**Frame Type**
Packet Length > Maximum Receive Buffer Size	No	None	No specific frame type
Packet Length 60 bytes to 1514 bytes (minimum size of the maximum receive buffer)	Yes	None	No specific frame type
Packet Length < 60 bytes	No	Short (bit 2)	No specific frame type
Packet Length > 1514 bytes	No	Long (bit 8)	No specific frame type

Note If the Monitor NLM supplies an ECB, the packet size is only limited by the adapter's ability to receive the packet.

Table 2
Ethernet Packet Length Field Rules

Ethernet Packet Length Field Rules			
Condition	**Deliver to Stack**	**Status Bit Set**	**Frame Type**
Length Field <= Packet Length – Header Length	Yes	None	No specific frame type
Packet Length > 21 AND Length Field > Packet Length –14	No	Malformed (bit 10)	No specific frame type
Packet Length >13 AND Length Field < Header Length –14	No	Malformed (bit 10)	No specific frame type

Table 3
Ethernet Frame Type Rules

Ethernet Frame Type Rules			
Condition	**Deliver to Stack**	**Status Bit Set**	**Frame Type**
Packet Length < 22 bytes	No	No Board (bit 9)	No specific frame type
Length Field <= 1500 AND Byte 14–15 : FF FF	Yes	None	802.3
Length Field <= 1500 AND Byte 14–16 : AA AA 03	Yes	None	SNAP
Length Field > 1500	Yes	None	EII
Length Field <= 1500 AND !Byte 14–15 : FF FF AND !Byte 14–16 : AA AA 03	Yes	None	802.2

Table 4
Ethernet CRC Rules

Ethernet CRC Rules			
Condition	**Deliver to Stack**	**Status Bit Set**	**Frame Type**
Good CRC	Yes	None	No specific frame type
Bad CRC	No	CRC (bit 0)	No specific frame type
Alignment Error	No	Alignment (bit 1)	No specific frame type
Bad CRC and Alignment Error	No	CRC and Alignment (bit 1)	No specific frame type

Token-Ring Packets

Token-Ring packets are created based upon the variants of source routing information, FC byte, packet size, CRC error status, and packet type.

Table 5
Token-Ring Packet Length Rules

Token-Ring Packet Length Rules			
Condition	**Deliver to Stack**	**Status Bit Set**	**Frame Type**
Packet Length > Maximum Receive Buffer Size	No	No change in status	No specific frame type

Note If the Monitor NLM supplies an ECB, the packet size is only limited by the adapter's ability to receive the packet.

Table 6
Token-Ring Source Route Field Rules

Token-Ring Source Route Field Rules			
Condition	**Deliver to Stack**	**Status Bit Set**	**Frame Type**
Packet Length < 802.5 header length + Source Route Length	No	No board (bit 9)	No specific frame type
Packet Length > 16 AND Source Route Bit is set	No	No change in status	No specific frame type
Packet Length >= 16 AND Source Route Length = 0	No	No board (bit 9)	No specific frame type
Packet Length >= 16 AND Source Route Length is odd	No	No board (bit 9)	No specific frame type

Table 7
Token-Ring Frame Type Rules

Token-Ring Frame Type Rules			
Condition	**Deliver to Stack**	**Status Bit Set**	**Frame Type**
Hdr-2 Length < 3 bytes AND LLC Packet	No	No Board (bit 9)	No specific frame type
Hdr-2 Length = 3 bytes AND LLC Packet AND !U frame	No	No Board (bit 9)	No specific frame type
Hdr-2 Length = 3 bytes AND LLC Packet AND U frame AND !Data Byte 0–2 : AA AA 03	Yes	No change in status	802.2
Hdr-2 Length = 3 to 7 bytes AND !Data Byte 0–2 : AA AA 03 AND LLC Packet	No	No Board (bit 9)	No specific frame type
Hdr-2 Length > 3 bytes AND Data Byte 0–2 : AA AA 03 AND LLC Packet	Yes	No change in status	802.2
Hdr-2 Length >= 8 bytes AND Data Byte 0–2 : AA AA 03 AND LLC Packet	Yes	No change in status	SNAP

Token-Ring Frame Type Rules *(continued)*			
Condition	**Deliver to Stack**	**Status Bit Set**	**Frame Type**
Hdr-2 Length < 4 bytes AND MAC Packet	No	No Board (bit 9)	No specific frame type
Hdr-2 Length >= 4 bytes AND MAC Packet	Yes	No change in status	No specific frame type
Frame type != LLC OR Frame Type != MAC	No	No board (bit 9)	No specific frame type

Table 8
Token-Ring CRC Rules

Token–Ring CRC Rules			
Condition	**Deliver to Stack**	**Status Bit Set**	**Frame Type**
Good CRC	Yes	No change in status	No specific frame type
Bad CRC	No	CRC (bit 0)	No specific frame type
Alignment Error	No	Alignment (bit 1)	No specific frame type
Bad CRC and Alignment Error	No	CRC and Alignment (bit 1)	No specific frame type

FDDI Packets

FDDI packets are created based upon the variants of source routing information, FC byte, packet size, CRC error status, and packet type.

Table 9
FDDI Packet Length Rules

FDDI Packet Length Rules			
Condition	**Deliver to Stack**	**Status Bit Set**	**Frame Type**
Packet Length > Maximum Receive Buffer Size	No	No change in status	No specific frame type
Packet Length > 4493 bytes	No	No change in status	No specific frame type

Note If the Monitor NLM supplies an ECB, the packet size is only limited by the adapter's ability to receive the packet.

Table 10
FDDI Source Route Field Rules

FDDI Source Route Field Rules			
Condition	**Deliver to Stack**	**Status Bit Set**	**Frame Type**
Packet Length < FDDI header length + Source Route Length	No	No board (bit 9)	No specific frame type
Packet Length > 15 AND Source Route Bit is set	No	No change in status	No specific frame type
Packet Length >= 15 AND Source Route Length = 0	No	No board (bit 9)	No specific frame type
Packet Length >= 15 AND Source Route Length is odd	No	No board (bit 9)	No specific frame type

Table 11
FDDI Frame Type Rules

FDDI Frame Type Rules			
Condition	**Deliver to Stack**	**Status Bit Set**	**Frame Type**
Hdr-2 Length < 3 bytes AND LLC Packet	No	No Board (bit 9)	No specific frame type
Hdr-2 Length = 3 bytes AND LLC Packet AND !U frame	No	No Board (bit 9)	No specific frame type
Hdr-2 Length = 3 bytes AND LLC Packet AND U frame AND !Data Byte 0–2 : AA AA 03	Yes	No change in status	802.2
Hdr-2 Length = 3 to 7 bytes AND !Data Byte 0–2 : AA AA 03 AND LLC Packet	No	No Board (bit 9)	No specific frame type
Hdr-2 Length > 3 bytes AND Data Byte 0–2 : AA AA 03 AND LLC Packet	Yes	No change in status	802.2
Hdr-2 Length >= 8 bytes AND Data Byte 0–2 : AA AA 03 AND LLC Packet	Yes	No change in status	SNAP
Hdr-2 Length < 4 bytes AND MAC Packet	No	No Board (bit 9)	No specific frame type
Hdr-2 Length >= 4 bytes AND MAC Packet	Yes	No change in status	No specific frame type
Frame type != LLC OR Frame Type != MAC	No	No board (bit 9)	No specific frame type

Table 12
FDDI CRC Rules

FDDI CRC Rules			
Condition	**Deliver to Stack**	**Status Bit Set**	**Frame Type**
Good CRC	Yes	No change in status	No specific frame type
Bad CRC	No	CRC (bit 0)	No specific frame type
Alignment Error	No	Alignment (bit 1)	No specific frame type
Bad CRC and Alignment Error	No	CRC and Alignment (bit 1)	No specific frame type

Receive Monitor Processor States

The processor states of the receive monitor routine are described below:

Entry State

EAX

contains the status of the packet (as follows):

0	=	a good packet
Bit 0 set	=	CRC Error
Bit 1 set	=	CRC/Alignment Error
Bit 2 set	=	Runt packet (Ethernet only)
Bit 8 set	=	Received packet is too big
Bit 9 set	=	No logical board is registered for this frame type
Bit 10 set	=	Malformed Packet (Ethernet only). Either the MAC header length is greater than the number of bytes read in, or the expected size of the MAC header is greater than the number of bytes read in.
Bits 11–30		Reserved.
Bit 31 set	=	MLID is shutting down.

EDI

has a pointer to a *LookAheadStructure.*

```
LookAheadStructure struc
  LMediaHeaderPtr   dd  0          ;Pointer to the media header
  LookAheadPtr      dd  0          ;Pointer to the look-ahead
                                   ;portion
  LookAheadLen      dd  0          ;Number of bytes in the
                                   ;look-ahead portion
  LProtID           db  6 dup (0)  ;Protocol ID
  LBoardNumber      dw  0          ;Board number of the packet
  LDataSize         dd  0          ;Total packet size.
LookAheadStructure ends
```

Note: If the *LookAheadLen* and the *LDataSize* fields are equal, the *LMediaHeaderPtr* and the *LookAheadPtr* fields have a pointer to a contiguous buffer that contains the entire packet. The receive monitor can copy the packet from this buffer.

Return State

EAX
has a 0 if the receive monitor must have more of the packet read in from the LAN adapter.

EBX
if EAX = 0, contains the number of bytes to skip from the beginning of the packet.

ECX
if EAX = 0, contains the number of bytes needed from the packet.

ESI
if EAX = 0, has a pointer to an ECB with a fragment structure that describes where the packet should be copied. (This ECB*'s RProtocolID* field should contain 'RCVMON'. Its *RESRAddress* field should contain the address of an ESR to call after the rest of the packet has been copied.

EDI
has a pointer to the ECB*'s RFragmentCount* field.

Preserved
EBP, and ESI (if EAX = 0).

Zero Flag
depends on EAX.

Receive Monitor Implementation Examples

If the MLID has already determined that it will pass the packet to the LSL in an ECB, it obtains the ECB, reads the packet in and calls the receive monitor. Because the entire packet is already there (*LDataSize = LookAheadLen)*, the receive monitor will not ask for more of the packet. The MLID then queues the packet on the LSL's hold queue by calling *LSLHoldRcvEvent*.

If the MLID will not pass the packet to the LSL (perhaps because the destination address didn't match or the Media header was not registered), it must pass the first *LookAheadSize* bytes of the packet to the receive monitor. The receive monitor is able to use *Ctl9_LookAheadSize* to define the look-ahead size. If the receive monitor must have more of the packet, the MLID must copy it from the LAN adapter and into the buffer returned in the ECB. The MLID passes the ECB to the LSL, and the LSL calls the ESR registered in the ECB.

If the MLID is driving a bus-mastering LAN adapter that preallocates ECBs and completely fills them in, the MLID gives the ECB to the receive monitor. Because the entire packet is already in the ECB, the receive monitor will not request more of the packet. The MLID then determines if it must pass the ECB to the LSL.

The receive monitor can also register as a transmit monitor when it calls *Ctl11_RegisterReceiveMonitor*. The transmit

monitor is passed a TCB (Transmit Control Block), and then must decide whether to copy part or all of the packet described by the TCB. The transmit monitor must not modify the packet. The transmit monitor has the following processor entry state:

ESI
has a pointer to a TCB.

The transmit monitor must preserve the ESI and EBP registers when it returns.

The Transmit Control Block (TCB)

The transmit monitor uses TCBs to transmit packets. The following section describes the TCB for Ethernet, Token-Ring and FDDI. RX-Net requires a different TCB structure which is also described.

The Ethernet, Token-Ring, and FDDI TCB

The following figures describe the Ethernet, Token-Ring, and FDDI TCB.

Figure 2 Transmit Control Block Source Code

```
TCBStructure    struc
    TCBDriverWS         dd  3    dup (4)
    TCBDataLen          dd  0
    TCBFragStrucPtr     dd  0
    TCBMediaHeaderLen   dd  0
TCBStructure    ends
TCBMediaHeader          equ      TCBMediaHeaderLen + 4
```

Note The fragment structure is defined on page

Figure 3 Graphic Representation of Transmit Control Block

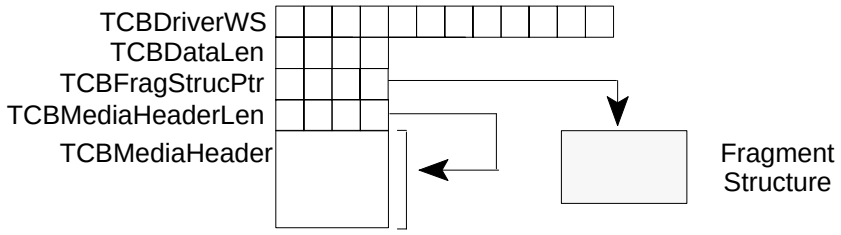


Table 13
Transmit Control Block Field Descriptions

Transmit Control Block Field Descriptions			
Offset	**Name**	**Bytes**	**Description**
00h	TCBDriverWS	12	The HSM may use this field for any purpose as long as it controls the TCB.
0Ch	TCBDataLen	4	This field contains the length of the packet described by the data fragments plus the media header. This value will never be 0.
10h	TCBFragStrucPtr	4	This field contains a pointer to a list of fragments defined by the *FragmentStructure* (described following the TCB section).
14h	TCBMediaHeaderLen	4	This field contains the length of the Media Header that immediately follows the TCB in memory. This value may be odd, even, or zero. A value of zero indicates a raw send. If the HSM is handed a raw send, the originating protocol stack has already included the media header in the first data fragment.
18h	TCBMediaHeader	?	Immediately following the TCB in memory is a buffer containing the Media Header that was assembled by the MSM.

RX-Net TCB

Because RX-Net handles fragmented packets differently and has different frame types, it requires a different TCB structure. The RX-Net TCB is described by the following figures.

Figure 4
RX-Net Transmit Control Block Source Code

```
TCBStructure    struc
    TCBDriverWS          dd      3    dup (4)
    TCBDataLen           dd      0
    TCBFragStrucPtr      dd      0
    TCBMediaHeaderLen    dd      0
TCBStructure    ends
TCBMediaHeader           equ     TCBMediaHeaderLen + 4
;The TCBMediaHeader contains:
    SourceAddress        db      ?
    DestinationAddress   db      ?
    LongPacketFlag       db      0
    ByteOffset           db      ?
    TCBSecondHeaderLen   db      ?
    Pad                  db      4    dup(?)  ;
    ProtocolType         db      ?
    SplitFlag            db      ?
    PacketSequence       dw      ?
```

Figure 5
Graphic Representation of RX-Net Transmit Control Block

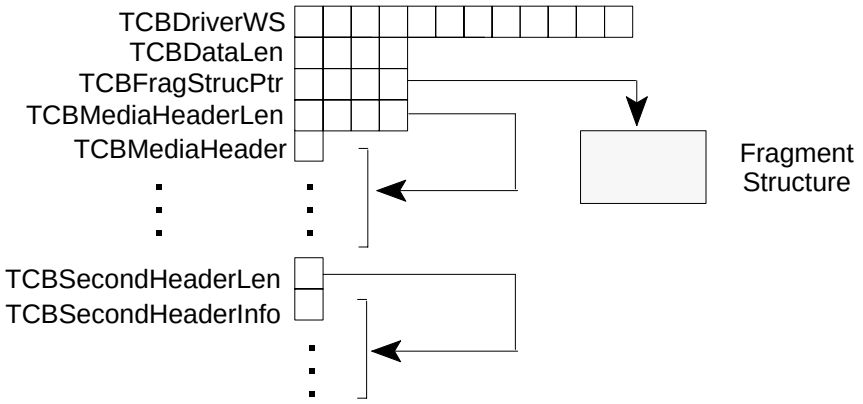


Table 14
RX-Net Transmit Control Block Field Descriptions

RX-Net Transmit Control Block Field Descriptions

Offset	Name	Bytes	Description
00h	TCBDriverWS	12	The MSM uses this field to link the TCBs
0Ch	TCBDataLen	4	This field contains the length of the packet described by the data fragments plus the media header. This value will never be 0.

RX-Net Transmit Control Block Field Descriptions
(continued)

Offset	Name	Bytes	Description
10h	TCBFragStrucPtr	4	This field contains a pointer to a list of fragments defined by the *FragmentStructure* (described following this section).
14h	TCBMediaHeaderLen	4	This field contains the length of the first media header.
Immediately following the TCB in memory is a buffer containing the media header information.			
18h	TCBMediaHeader	3 or 4	This field contains the first media header. The header is 3 bytes for Short Packets and 4 bytes for Long or Exception Packets.
?	TCBSecondHeaderLen	1	This field contains the length of the second media header.
?	TCBSecondHeader	4 or 8	This field contains the second media header. The header is 4 bytes for Short or Long Packets, and 8 bytes for Exception Packets.

The Fragment Structure

This structure contains the number and location of data fragments.

Figure 6
FragmentStructure
Source Code

```
    FragmentCount       dd      0
    FragmentOffset1     dd      0
    FragmentLength1     dw      0
                .
                .
                .
;Additional fragments for FragmentCount>1
    FragmentOffsetn     dd      0
    FragmentLengthn     dw      0
```

Figure 7
Graphic Representation of the *FragmentStructure*

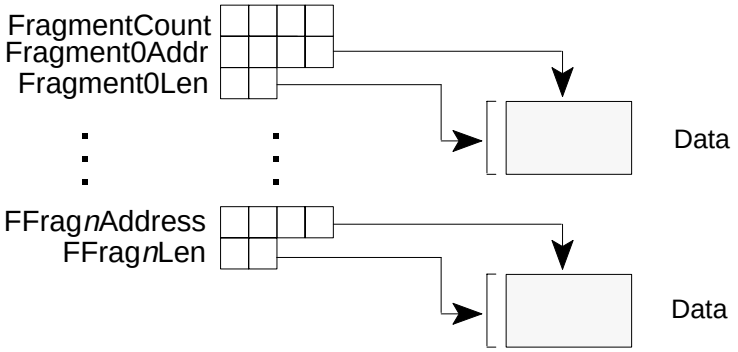


Table 15
Description of *FragmentStructure* Fields

Description of *FragmentStructure* Fields			
Size	**Name**	**Bytes**	**Description**
00h	FragmentCount	4	This field contains the number of data fragment descriptors to follow. Each descriptor consists of a pointer to a fragment buffer and the size of that buffer. The HSM collects the data from these buffers when forming the packet for transmission.
04h 08h . . .	FragmentOffset1 FragmentLength1 . . . FragmentOffsetn FragmentLengthn	4 4 . . .	Pointer to the buffer containing the first data fragment. Length of the buffer pointed to by *FragmentOffset1*. (These fields contain additional fragment descriptors)

Ctl11_RegisterReceiveMonitor

Description

Allows protocol stacks to monitor all received packets.

Entry State

EAX
has the logical board number.

EBX
subfunction = 11.

ECX
equals 0 to deregister.
nonzero to register.

ESI
has a pointer to the receive monitor routine (can be 0).

EDI
has a pointer to the transmit monitor routine (can be 0).
Note: Either ESI or EDI or both must be nonzero. A routine pointer that is 0 will not be called.

If unloading a receive monitor:
EDX
contains the chain ID.

Return State

EAX
has a completion code.

Completion Codes (EAX)

00000000h *Successful*

0FFFFF81h *BadCommand*
The MLID does not support receive monitoring.

0FFFFF84h *Failure*
The routine is trying to disable a receive monitor or a transmit monitor that is not registered to this MLID.

0FFFFF89h *OutOfResources*
Another receive monitor is already registered.

Description

Ctl11_RegisterReceiveMonitor allows protocol stacks to monitor all packets that the LAN adapter receives. Protocol stacks also use this routine to enable and disable the receive monitor registered to a LAN adapter.

The receive monitor examines all packets sent from or received by a LAN adapter. If promiscuous mode is supported, the

Ctl11_RegisterReceiveMonitor *continued*

receive monitor can request that the adapter enter promiscuous mode. When promiscuous mode is enabled, the MLID should allow all packets (including bad packets, if possible) to be passed up to the receive monitor function. Only one receive monitor function at a time can be registered with an MLID.

Index