TSM Procedures

Introduction 6-2

<TSM>GetNextSend 6-3
<TSM>GetRCB 6-4
<TSM>ProcessGetRCB 6-6
<TSM>FastProcessGetRCB 6-8
<TSM>RcvComplete 6-10
<TSM>FastRcvComplete 6-11
<TSM>RegisterHSM 6-12
<TSM>SendComplete 6-13
<TSM>FastSendComplete 6-14
<TSM>UpdateMulticast 6-15

RXNetTSMGetRCB 6-16
RXNetTSMRcvEvent 6-19
RXNetTSMFastRcvEvent 6-20

Introduction

This chapter describes the topology specific procedures provided as tools for HSM developers.

The Topology Specific Module, <TSM>.NLM, manages the operations that are unique to a specific media type. Multiple frame support is implemented in the Topology Module so that all frame types for a given media are supported.

The topology specific functions are indicated with <TSM>. The developer must replace <TSM> with the appropriate media type depending on which module is used. Since the driver must be assembled with case sensitivity on, the names must be used exactly as shown.

```
ETHERTSM.NLM  ‣  replace <TSM> with:  EtherTSM
TOKENTSM.NLM  ‣  replace <TSM> with:  TokenTSM
RXNETTSM.NLM  ‣  replace <TSM> with:  RXNetTSM
PCN2LTSM.NLM  ‣  replace <TSM> with:  PCN2LTSM
 FDDITSM.NLM  ‣  replace <TSM> with:  FDDITSM
```

RX-Net drivers require special consideration to handle split packets. Several routines are provided that are specific to the RX-Net module. These routines are described at the conclusion of this chapter.

<TSM>GetNextSend

On Entry

EBP	Pointer to the Adapter Data Space
Interrupts	are disabled
Call	at process or interrupt time

On Return

EBP	Pointer to the Adapter Data Space
EBX	Pointer to the Frame Data Space
ESI	Pointer to the next TCB to transmit if successful. This routine will decrement *MSMTxFreeCount*.
ECX	Padded packet length (Ethernet only)
Zero Flag	Set if successful; otherwise there are no TCBs queued or the adapter is currently using all of its transmit resources and cannot accept another packet *(MSMTxFreeCount = 0).*
Interrupts	are disabled

Description

This function retrieves the next ECB to be sent from the MSM's transmit waiting queue. It then builds a TCB and gives it to the HSM for transmission. If the send queue is empty, this function clears the zero flag and returns.

Note: The *DriverSend* routine may use ECBs instead of TCBs by initializing the *DriverParameterBlock* variable *DriverSendWantsECBs* to a non-zero value (see Chapter 3). In this case, *<TSM>GetNextSend* will simply retrieve the next ECB to be sent (without building a TCB).

Example

```
DriverISR   proc
     •
     •
     •
TransmitComplete:                     ; EBP = Ptr to Adapter Data Space

   inc    [ebp].MSMTxFreeCount        ; Free adapter's transmit resource
   mov    [ebp].TxInProgress, 0       ; Clear transmit in progress flag

   ;*** Transmit Next Packet ***

   call   <TSM>GetNextSend            ; Get the next TCB from the queue
   jnz    NoSendsQueued               ; Jump if nothing to send
   call   DriverSend                  ; Otherwise send the packet
     •
     •
     •
DriverISR   endp
```

<TSM>GetRCB
(For RX-Net see "RXNetTSMGetRCB")

On Entry

EBP	Pointer to the Adapter Data Space
ESI	Pointer to the received packet header (LookAhead buffer)
ECX	Size of the received packet
EAX	Status of received packet for the Receive Monitor (see *DriverPromiscuousChange* in Chapter 5)
Interrupts	can be in any state
Call	at process or interrupt time

On Return

Zero Flag	Set if successful; otherwise an error occurred
ESI	Pointer to the fragmented RCB if this call is successful
EDI	Pointer to the fragment structure (points to the *RCBFragmentCount* field of the RCB)
EBX	Number of bytes to skip over from the beginning of packet
ECX	Number of bytes remaining to read
Interrupts	are disabled
Note	EBP is preserved

Description

This routine is called by the HSM to obtain a fragmented RCB for a packet that has been received by the adapter. Drivers that cannot handle fragmented receive buffers should obtain RCBs using either *MSMAllocateRCB* or *<TSM>ProcessGetRCB*.

<TSM>GetRCB uses a LookAhead process in which the packet's header information is previewed before an RCB is given to the driver. This way the TSM can first verify that it wants the packet, before the driver transfers the entire packet from the adapter into an RCB.

The adapter's data transfer method governs how the LookAhead process is handled.

- If a programmed I/O adapter is being used, the HSM must transfer the packet's header information from the adapter into a buffer maintained for this purpose. The number of bytes to transfer is specified by the variable *MSMMaxFrameHeaderSize* described in Chapter 4. The HSM must set ESI to point the beginning of the LookAhead buffer before calling this routine.

- If a shared RAM (memory-mapped I/O) adapter is being used, the HSM can simply point ESI to the beginning of the packet buffer in shared RAM.

On entry to this routine, ESI must point to the packet's header information (the LookAhead buffer) and ECX must contain the size of the received packet. If the header verifies, the TSM will obtain an RCB and use the LookAhead information to fill in the *RCBReserved* fields before returning a pointer to the RCB in ESI.

After obtaining the RCB, the remainder of the packet must be transferred into the RCB fragment buffers. EBX is the offset from the beginning of the packet to start copying from and ECX contains the number of bytes in the packet left to read.

After the HSM has read the rest of the packet, it must return the RCB to the LSL using either the *<TSM>RcvComplete* / *MSMServiceEvents* combination or by using *<TSM>FastRcvComplete*.

Note: If this routine returns an error completion code, the received packet should be discarded.

Special Instructions **Bus Master Adapters**
Bus Master devices require RCBs to be preallocated. Since this routine requires a LookAhead Buffer, preallocation is not possible. The HSM can preallocate RCBs using the macro *MSMAllocateRCB*.

Example

```
mov   ecx, [ebp].MSMMaxFrameHeaderSize ; Build LookAhead buffer
lea   edi, [ebp].LookAheadBuffer
rep   insb

lea   esi, [ebp].LookAheadBuffer       ; Ptr to LookAhead buffer
mov   ecx, PacketSize                  ; Size of the received packet
call  <TSM>GetRCB                      ; Get an RCB
jnz   PacketNotAccepted                ; Jump if Error
 •
 •    (Copy remainder of the packet into the RCB)
 •
call  <TSM>RcvComplete                 ; Return RCB
```

<TSM>ProcessGetRCB

(For RX-Net see "RXNetTSMRcvEvent")

On Entry

EBP	Pointer to the Adapter Data Space
ESI	Pointer to the received packet's RCB
ECX	Size of the received packet
EAX	Status of received packet for the Receive Monitor (see *DriverPromiscuousChange* in Chapter 5)
EDI	Maximum packet size for the new RCB
Interrupts	can be in any state
Call	at process or interrupt time

On Return

Zero Flag	Set if a new RCB was available
ESI	Pointer to a new non-fragmented RCB (if the zero flag is set)
Interrupts	are disabled
Note	EBP is preserved

Description

The HSM calls this routine to process an RCB for a received packet and to preallocate a new non-fragmented RCB for the next packet. The received packet must have been copied into the *RCBDataBuffer*.

Use this routine if the RCB was preallocated using *MSMAllocateRCB* or was obtained from a previous call to this routine. In either case, the *RCBReserved* fields **have not** been filled in, and therefore must be completed by the TSM.

Note: If the adapter/driver is "ECB aware" and has already filled in all required ECB fields as described in Chapter 4, the ECB should be returned for processing using *<TSM>RcvComplete / MSMServiceEvents* or *<TSM>FastRcvComplete*.

When this routine is called, the TSM examines the packet header information. If the header verifies, the *RCBReserved* fields are filled in and the RCB is directed to the Link Support Layer's holding queue to await processing. The TSM then obtains a new non-fragmented RCB, if one is available, and returns it to the driver. If the packet header is invalid, the RCB will be given back to the driver to be used again for another packet.

The HSM must eventually use the macro *MSMServiceEvents* which enables the RCB's Event Service Routine to complete the processing.

Special Instructions

Ethernet
The HSM should start copying the packet from the 6 byte destination field of the media header into the *RCBDataBuffer* field of the RCB.

Token-Ring
The HSM should start copying the packet from the Access Control byte of the media header into the *RCBDataBuffer* field of the RCB.

FDDI
The HSM should start copying the packet from the Frame Control byte of the media header into the *RCBDataBuffer* field of the RCB.

PCN2L
This routine is not available if using the PCN2L TSM.

Example

```
DriverInit  proc
   ⋮
   mov   esi, [ebx].MLIDMaximumSize
   call  MSMAllocateRCB                 ; Preallocate first RCB and save.
   ⋮
DriverInit  endp



DriverISR   proc
   ⋮
ReceiveEvent:
   ⋮  (Copy packet into the RCBDataBuffer field of the preallocated RCB)
   xor   eax, eax                       ; Good packet
   mov   ecx, PacketSize                ; Size of received packet
   mov   edi, [ebx].MLIDMaximumSize     ; Maximum size for new RCB
   call  <TSM>ProcessGetRCB             ; Return RCB and get a new RCB
   ⋮
DriverISR   endp
```

<TSM>FastProcessGetRCB

(For RX-Net see "RXNetTSMFastRcvEvent")

On Entry

EBP	Pointer to the Adapter Data Space
ESI	Pointer to the received packet's RCB
ECX	Size of the received packet
EAX	Status of received packet for the Receive Monitor (see *DriverPromiscuousChange* in Chapter 5)
EDI	Maximum packet size for the new RCB
Interrupts	can be in any state (but might be enabled during the call)
Call	at process or interrupt time

On Return

Zero Flag	Set if a new RCB was available
ESI	Pointer to a new non-fragmented RCB (if the zero flag is set)
Interrupts	are disabled
Note	EBP is preserved

Description

<TSM>FastProcessGetRCB is identical to *<TSM>ProcessGetRCB* with the exception that before this routine returns, the RCB's Event Service Routine is called to complete the processing. *<TSM>ProcessGetRCB* used in conjunction with *MSMServiceEvents* will perform the same task.

During the RCB's Event Service Routine, the interrupts might be enabled and all registers could be destroyed. The HSM should preserve any needed registers before calling *<TSM>FastProcessGetRCB*. If having the interrupts enabled is undesirable, the driver should use the *<TSM>ProcessGetRCB* procedure and wait until the conclusion of the receive routine before servicing events.

It is also important to note that the HSM's *DriverSend* routine may be called before this procedure returns. This can be prevented by starting a critical section before calling this routine.

Caution: This routine calls the RCB's Event Service Routine during which the interrupts might be enabled and all registers could be destroyed.

This routine is not available if using the PCN2L TSM.

Example

```
DriverInit  proc
   ⋮
   mov   esi, [ebx].MLIDMaximumSize
   call  MSMAllocateRCB                    ; Preallocate first RCB and save.
   ⋮
DriverInit  endp



DriverISR   proc
   ⋮
ReceiveEvent:
   ⋮  (Copy packet into the RCBDataBuffer field of the preallocated RCB)
   xor   eax, eax                          ; Good packet
   mov   ecx, PacketSize                   ; Size of received packet
   mov   edi, [ebx].MLIDMaximumSize        ; Maximum size for new RCB
   call  <TSM>FastProcessGetRCB            ; Return RCB, service events, and
   ⋮                                       ; get a new RCB
DriverISR   endp
```

<TSM>RcvComplete

On Entry

EBP	Pointer to the Adapter Data Space
ESI	Pointer to the received packet's RCB
Interrupts	are disabled
Call	at process or interrupt time

On Return

Interrupts	are disabled
Note	EBP, ESI, and EDI are preserved.

Description

The HSM calls *<TSM>RcvComplete* to direct a completed RCB to the Link Support Layer's holding queue to await processing. Use this routine if the RCB was obtained using the *<TSM>GetRCB* procedure and the received packet has been copied into the RCB receive buffer(s).

Note: This procedure is also used if the adapter/driver is *ECB aware* and has filled in all required ECB fields as described in Chapter 4.

When an RCB is queued using this routine, the HSM must eventually use the macro *MSMServiceEvents* to call the RCB's Event Service Routine and complete the processing.

Special Instructions

RX-Net
If an RX-Net Adapter/Driver is *ECB aware* (see Chapter 4), it is responsible for handling packet reconstruction and fragmentation. Once the packet is reconstructed, the HSM must set the second byte of the *DriverWorkspace* field to one before calling this routine.

Example

```
mov   ecx, [ebp].MSMMaxFrameHeaderSize ; Build the LookAhead buffer
lea   edi, [ebp].LookAheadBuffer
rep   insb

lea   esi, [ebp].LookAheadBuffer       ; Ptr to LookAhead buffer
mov   ecx, PacketSize                  ; Size of the received packet
call  <TSM>GetRCB                      ; Get an RCB
jnz   PacketNotAccepted                ; Jump if Error
    •
    •  (Copy remainder of the packet into the RCB)
    •
call  <TSM>RcvComplete                 ; Return the RCB
```

<TSM>FastRcvComplete

On Entry

EBP	Pointer to the Adapter Data Space
ESI	Pointer to the received packet's RCB
Interrupts	are disabled (but might be enabled during the call)
Call	at process or interrupt time

On Return

Interrupts	are disabled
Note	Assume all registers are destroyed.

Description

<TSM>FastRcvComplete is identical to *<TSM>RcvComplete* with the exception that before this routine returns, the RCB's Event Service Routine is called to complete the processing. Using *<TSM>RcvComplete* in conjunction with *MSMServiceEvents* will perform the same task.

During the RCB's Event Service Routine, the interrupts might be enabled and all registers could be destroyed. The HSM should preserve any needed registers before calling *<TSM>FastRcvComplete*. If having the interrupts enabled is undesirable, the driver should use the *<TSM>RcvComplete* procedure and wait until the conclusion of the receive routine before servicing events.

It is also important to note that the HSM's *DriverSend* routine may be called before this procedure returns. This can be prevented by starting a critical section before calling this routine.

Caution: This routine calls the RCB's Event Service Routine during which the interrupts might be enabled and all registers could be destroyed.

Example

```
mov   ecx, [ebp].MSMMaxFrameHeaderSize ; Build the LookAhead buffer
lea   edi, [ebp].LookAheadBuffer
rep   insb

lea   esi, [ebp].LookAheadBuffer       ; Ptr to LookAhead buffer
mov   ecx, PacketSize                  ; Size of the received packet
call  <TSM>GetRCB                      ; Get an RCB
jnz   PacketNotAccepted                ; Jump if Error
    •
    •  (Copy remainder of the packet into the RCB)
    •
call  <TSM>FastRcvComplete             ; Return RCB and service events
```

<TSM>RegisterHSM

On Entry

ESI	Pointer to the DriverParameterBlock structure
Interrupts	can be in any state
Call	at initialization time only

On Return

EAX	Zero if successful; otherwise EAX points to an error message that the driver must print using *MSMPrintString* before returning to the operating system with EAX non-zero.
EBX	Pointer to the Frame Data Space
Interrupts	are disabled
Note	All other registers are destroyed
Zero Flag	Set if successful; otherwise an error occurred

Description

The HSM's *DriverInit* routine must call *<TSM>RegisterHSM* with a pointer to its *DriverParameterBlock* structure in ESI. Before calling this routine, *DriverInit* must save the value of the stack pointer in the *DriverStackPointer* field of the *DriverParameterBlock* after pushing the C registers EBP, EBX, ESI, and EDI. This routine then calls the MSM which performs the following tasks:

- copies the parameter block into local data space
- processes driver firmware variables
- allocates the Frame Data Space
- copies the driver configuration table into the Frame Data Space
- parses information derived from the linker definition file
- places LSL's maximum packet size in the configuration table
- initializes screen ID used for *MSMPrintString* procedures

Example

```
DriverInit  proc
   Cpush                           ; macro to save "C" registers
   mov   DriverStackPointer, esp   ; Fill in stack pointer
   lea   esi, DriverParameterBlock ; Get pointer to Parameter Block
   call  <TSM>RegisterHSM          ; Get a Frame Data Space
   jnz   DriverInitError           ; Jump if error
     ⋮
   xor   eax, eax                  ; Successful return with EAX=0
   Cpop                            ; Restore "C" registers
   ret


DriverInitError:
   mov   esi, eax                  ; ESI=EAX= ptr to error message
   call  MSMPrintString            ; Print the Message
   Cpop                            ; Restore "C" Registers
   ret                             ; Return (EAX is non-zero on errors)
DriverInit  endp
```

<TSM>SendComplete

On Entry

EBP	Pointer to the Adapter Data Space
ESI	Pointer to the Transmit Control Block (TCB)
Interrupts	are disabled
Call	at process or interrupt time

On Return

Interrupts	are disabled
Note	EBP is preserved.

Description

This procedure is called to release a TCB after a packet has been transmitted. It can be called by *DriverISR* after a transmit complete interrupt or by the *DriverSend* routine before the actual transmission is complete (a "lying send"), as long as all packet data has been transferred into the adapter's transmit buffer and access to the TCB is no longer required.

This procedure returns the packet's TCB to the MSM's unused TCB queue and directs the underlying Transmit ECB to the Link Support Layer's service queue.

The HSM must eventually use the macro *MSMServiceEvents* which calls the ECB's Event Service Routine. Typically, if the *DriverSend* routine was called to transmit the next packet after a send complete interrupt, then the interrupt service routine should invoke *MSMServiceEvents*.

Note: The *DriverSend* routine may use ECBs instead of TCBs by initializing the Driver Parameter Block variable *DriverSendWantsECBs* to a non-zero value (see Chapter 3). In this case, *<TSM>SendComplete* will simply direct the ECB to the LSL's service queue.

Example

```
DriverSend   proc
   •
   •   (send the packet to the NIC)
   •
   cmp   InDriverISR, 0
   jnz   <TSM>SendComplete
   jmp   <TSM>FastSendComplete

DriverSend    endp
```

<TSM>FastSendComplete

On Entry

EBP	Pointer to the Adapter Data Space
ESI	Pointer to the Transmit Control Block (TCB)
Interrupts	are disabled (but might be enabled during the call)
Call	at process or interrupt time

On Return

Interrupts	are disabled
Note	All registers are destroyed.

Description

<TSM>FastSendComplete is identical to *<TSM>SendComplete* with the exception that before this routine returns, the TCB's Event Service Routine is called to notify the upper layers that the transmission is complete. Using the *<TSM>SendComplete* / *MSMServiceEvents* combination will perform the same task.

During the TCB's Event Service Routine, the interrupts might be enabled and all registers could be destroyed. The HSM should preserve any needed registers before calling *<TSM>FastSendComplete*. It is also important to note that the HSM's *DriverSend* routine may be called before this procedure returns. This can be prevented by starting a critical section before calling this routine.

Example

```
DriverSend  proc
   •
   •  (send the packet to the NIC)
   •
   cmp   InDriverISR, 0
   jnz   <TSM>SendComplete
   jmp   <TSM>FastSendComplete

DriverSend  endp
```

<TSM>UpdateMulticast

On Entry

EBP	Pointer to the Adapter Data Space
Interrupts	are disabled and remain disabled
Call	at process or interrupt time

On Return

Interrupts	are disabled
Note	EBX and EBP are preserved

Description

When this routine is called it passes the current multicast table (maintained by the MSM) to the HSM's *DriverMulticastChange* routine. This allows the driver to update the adapter's multicast address registers.

This routine is called by internal TSM procedures each time the multicast addresses are added to or deleted from the MSM's multicast table. This routine can also be called by the driver during the HSM's *DriverReset* routine.

Note: RX-Net does not support multicast addressing. This routine is not available if the RXNetTSM module is used.

See Also

Refer to the sections covering the following flags and variables for more information on multicast addressing:

- Bit 3 of the *MLIDModeFlags* is used to indicate whether or not multicast addressing is supported.
- Bits 9 and 10 of the *MLIDFlags* must be set appropriately to reflect the multicast mechanism or format used by the adapter/driver.
- The *DriverParameterBlock* variable, *DriverMaxMulticast*, must be set to reflect the maximum number of multicast addresses the adapter can handle.

Example

```
DriverReset proc
   ⋮
   call  <TSM>UpdateMulticast
   ⋮
DriverReset endp
```

RXNetTSMGetRCB

On Entry

EBP	Pointer to the Adapter Data Space
ESI	Pointer to the LookAhead Buffer
Interrupts	are in any state
Call	at process or interrupt time

On Return

Zero flag	Set if successful; otherwise an error occurred
ESI	Pointer to the RCB if this call is successful
EDI	Pointer to the RCB fragment structure (points to the *RCBFragmentCount* field of the RCB)
EBX	contains the offset in the card's buffer from which to start copying data
ECX	Number of bytes remaining to read
Interrupts	are disabled
Note	EBP is preserved

Description

This routine is normally used for programmed I/O adapters.

RXNetTSMGetRCB uses a LookAhead process in which the packet's header information is previewed before an RCB is given to the driver. This way the TSM can first verify that it wants the packet, before the driver transfers the entire packet from the adapter into an RCB.

The LookAhead process requires the HSM to build a buffer containing the packet's header information as shown in Figure 6.1 on the following page. The number of bytes required for the buffer is specified by the variable *MSMMaxFrameHeaderSize* described in Chapter 4. The HSM must set ESI to point to the beginning of the LookAhead buffer before calling this routine. If the header verifies, this routine returns a pointer to an RCB in ESI.

At this point, the HSM must transfer the remainder of the packet into the RCB fragment structure. Since other fragments of a split packet may have already been copied into the RCB buffers, the HSM must perform the following operations.

- The dword value at [EDI – 4] indicates the number of bytes currently in the RCB fragment buffers. This value can be used along with the *RCBFragmentLength* fields to determine where in the RCB fragment structure to begin copying the packet.

- Once the position is located, the HSM transfers the rest of the packet into the RCB fragment structure. (EBX is the offset from the beginning of the card's buffer to start copying from and ECX is the number of bytes left to read.)

- Update the number of bytes currently in the RCB fragment buffers by adding ECX bytes to the dword value at [EDI – 4].

After the HSM completes the above tasks, it must return the RCB using either the *<TSM>RcvComplete* / *MSMServiceEvents* combination or by using *<TSM>FastRcvComplete*.

Note: Using *RXNetTSMGetRCB* does not provide 100% support to a receive monitor.

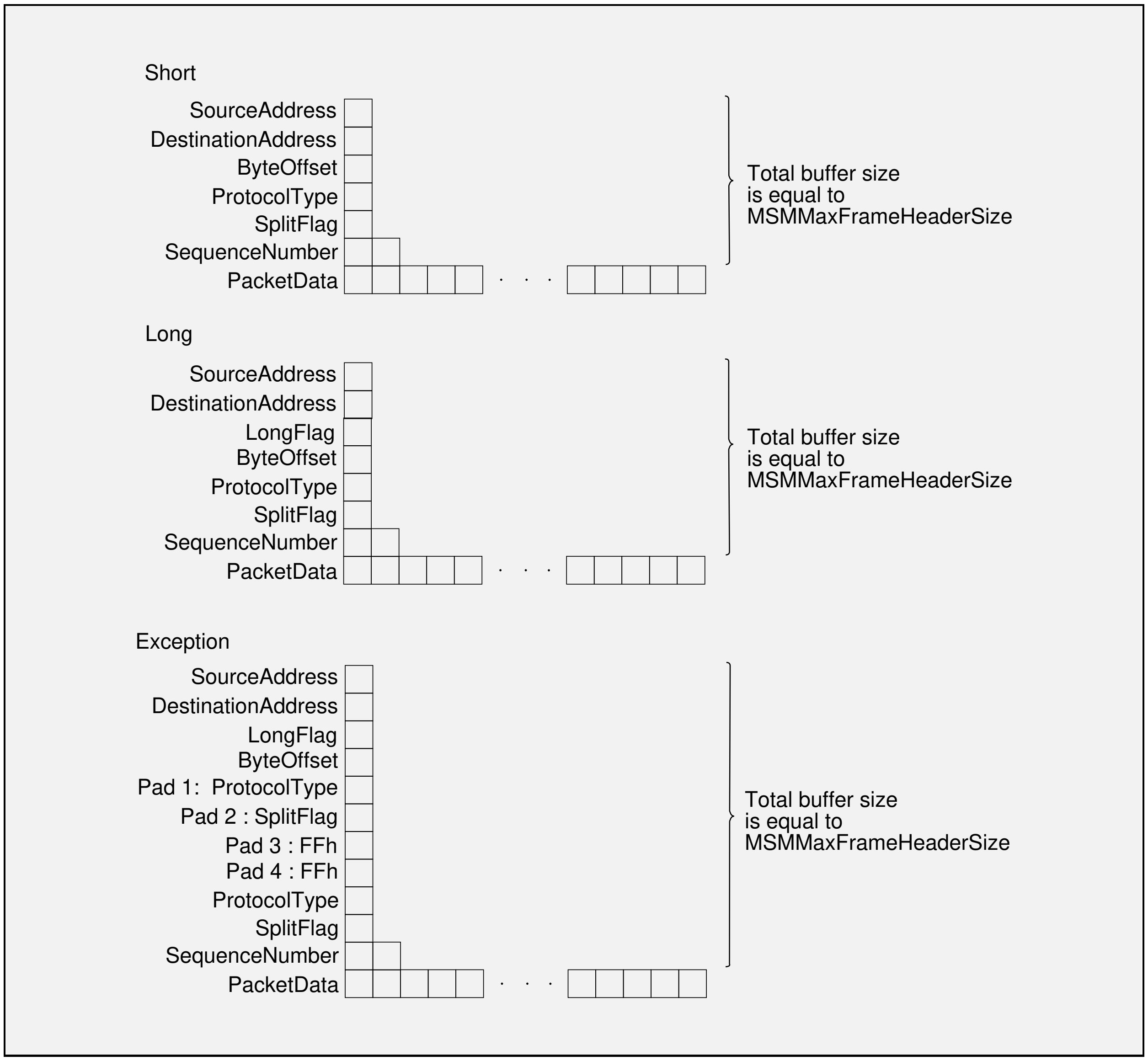

Figure 6.1 Format of RX-Net LookAhead Buffer

Example

```
 •
 •   (Build the LookAheadBuffer)
 •
lea   esi,[ebp].LookAheadBuffer
call  RXNetTSMGetRCB                   ; Get an RCB
jnz   NoRCB                            ; Jump if there is an error
 •
 •   (Determine the current fragment buffer position)
 •   (Transfer the rest of the packet into the RCB)
 •
add   [edi-4], ecx
call  RXNetTSMRcvComplete              ; Return the RCB
```

RXNetTSMRcvEvent
(RX-Net only)

On Entry

EBP	Pointer to the Adapter Data Space
ESI	Pointer to the received packet
Interrupts	are in any state
Call	at process or interrupt time

On Return

Zero Flag	set if successful
Interrupts	are disabled
Note	EBP is preserved

Description

RXNetTSMRcvEvent is only available to HSMs that use RX-Net shared RAM cards and that use the RXNETTSM module. The only action the HSM takes when a packet is received is to pass this routine a pointer to the packet. If the packet is wanted, the TSM copies the entire packet into an RCB, completing packet reception.

The HSM must eventually use the macro *MSMServiceEvents* which enables the RCB's Event Service Routine to complete the processing.

RX-Net cards that do not support shared RAM should either:

- Use the *RXNetTSMGetRCB* / *<TSM>RcvComplete* combination to receive packets. This method does not provide 100% support to a receive monitor.
- Copy the entire packet from the adapter into a buffer and call this routine with a pointer to that buffer in ESI. This method is the only way to provide 100% support to a receive monitor.

Example

```
mov   esi, [ebp].CurrRxPage      ; ESI -> Current Rx Page
xor   [ebp].CurrRxPage, 0200h    ; Toggle to the next page
call  RXNetTSMRcvEvent           ; Pass the packet to MSM
jmp   ISRExit                    ; Finished receiving the packet
```

RXNetTSMFastRcvEvent
(RX-Net only)

On Entry

EBP	Pointer to the Adapter Data Space
ESI	Pointer to the received packet
Interrupts	are in any state, but may be enabled
Call	at process or interrupt time

On Return

Zero Flag	set if successful
Interrupts	are disabled
Note	EBP is preserved

Description

RXNetTSMFastRcvEvent is identical to *RXNetTSMRcvEvent* with the exception that before this routine returns, the RCB's event service routine is called to complete the processing. Using *RXNetTSMGetRCB* / *RXNetTSMRcvEvent* in conjunction with *MSMServiceEvents* will perform the same task.

During the RCB's Event Service Routine, the interrupts might be enabled and all registers could be destroyed. The HSM should preserve any needed registers before calling *RXNetTSMFastRcvEvent*. If having the interrupts enabled is undesirable, the driver should use the *RXNetTSMRcvEvent* procedure and wait until the conclusion of the receive routine before servicing events.

It is also important to note that the HSM's *DriverSend* routine may be called before this procedure returns. This can be prevented by starting a critical section before calling this routine.

Caution: This routine calls the RCB's Event Service Routine, during which the interrupts might be enabled and all registers could be destroyed.

Example

```
mov   esi, [ebp].CurrRxPage      ;  location of received packet
call  RXNetTSMFastRcvEvent
```