LAN VCI Specification

Revision/Update Information: Version 1.0

OpenVMS Development

April 1, 1993

Abstract

This document describes the VMS Communications Interface (VCI) to the LAN device drivers on VMS.

Digital Equipment Corporation
Maynard, Massachusetts

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Restricted Rights: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

© Digital Equipment Corporation 1992.

All Rights Reserved.
Printed in U.S.A.

The postpaid Reader's Comments forms at the end of this document request your critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem–10	PDP	VT
DECSYSTEM–20	PDT	
DECUS	RSTS	
DECwriter	RSX	digital™

This document was prepared using VAX DOCUMENT, Version 2.1.

Contents

CHAPTER 1 INTRODUCTION 1–1

1.1 CONTENTS 1–1

1.2 PERFORMANCE 1–1

1.3 SUPPORT OBJECTIVES 1–1

CHAPTER 2 VCI OVERVIEW 2–1

2.1 TERMS 2–1
2.1.1 VMS COMMUNICATION MODULE 2–1
2.1.2 SERVICE ROUTINE 2–2
2.1.3 VCI PORT 2–2

2.2 OVERVIEW OF DATA STRUCTURES 2–2
2.2.1 LAN BLOCK 2–3
2.2.2 LAN DEVICE CHARACTERISTICS 2–3
2.2.3 VCI BLOCK 2–3
2.2.4 VMS COMMUNICATIONS REQUEST PACKET 2–3
2.2.5 DATA CHAIN BLOCK 2–3

2.3 OVERVIEW OF ROUTINES 2–3
2.3.1 ACCESS ROUTINES 2–4
2.3.2 PORT MANAGEMENT SERVICE 2–4
2.3.3 TRANSMIT SERVICE 2–4
2.3.4 RECEIVE SERVICE 2–5
2.3.5 REPORT EVENT SERVICE 2–5

CHAPTER 3 ROUTINE DESCRIPTIONS 3–1

3.1 ACCESS ROUTINES 3–1
3.1.1 LAN$GET_DEVICE 3–1
3.1.2 VCI$LAN_CREATE_PORT 3–2
3.1.3 VCI$LAN_DELETE_PORT 3–4

3.2 PORT MANAGEMENT SERVICE ROUTINES 3–5
3.2.1 VCI$LAN_PORTMGMT_INITIATE 3–5
3.2.2 VCI$xxx_PORTMGMT_COMPLETE 3–6

3.3 TRANSMIT SERVICE ROUTINES 3–6
3.3.1 VCI$LAN_TRANSMIT_INITIATE 3–8
3.3.2 VCI$LAN_BUILD_HDR 3–8
3.3.3 VCI$LAN_TRANSMIT_FRAME 3–9
3.3.4 VCI$xxx_TRANSMIT_COMPLETE 3–9

3.4 RECEIVE SERVICE ROUTINES 3–10
3.4.1 VCI$xxx_RECEIVE_COMPLETE 3–10

3.5 REPORT EVENT SERVICE ROUTINES 3–11
3.5.1 PORT USABLE EVENT 3–11

3.5.2 PORT UNUSABLE EVENT 3–12
3.5.3 RECEIVE CONGESTION EVENT 3–12
3.5.4 NEW ADDRESS EVENT 3–12
3.5.5 RECEIVE PDU LOST EVENT 3–13
3.5.6 RESTART FAILED EVENT 3–13
3.5.7 STATION RENAMED EVENT 3–13

3.6 STATUS VALUES 3–14

CHAPTER 4 DATA STRUCTURES 4–1

4.1 LAN BLOCK 4–1

4.2 LDC 4–2

4.3 VCIB 4–3

4.4 VCRP 4–9

4.5 DCBE 4–15

4.6 LIL 4–17

CHAPTER 5 FEATURES 5–1

5.1 PERFORMANCE 5–1

5.2 CONVERTING A VCRP INTO A DCBE 5–1

5.3 CONVERTING A VCRP INTO AN ACB 5–2

CHAPTER 6 PROCESSING 6–1

6.1 STARTING A VCI PORT 6–1

6.2 CHANGING A VCI PORT 6–2

6.3 SHUTTING DOWN A VCI PORT 6–3

6.4 TRANSMITTING A PACKET 6–3
6.4.1 TRANSMIT FRAME 6–3
6.4.2 TRANSMIT CHAINING 6–5

6.5 RECEIVING A PACKET 6–7

6.6 GETTING AN EVENT 6–8
6.6.1 PORT USABLE AND PORT UNUSABLE 6–8
6.6.2 NEW ADDRESS 6–8
6.6.3 RESTART FAILED 6–9

APPENDIX A VCI USER IDS A–1

APPENDIX B REVISION HISTORY B–1

TABLES

3–1 Create port status values 3–3
3–2 Delete port status values 3–4
3–3 Port management request status values 3–6
3–4 Transmit request status values 3–10
3–5 Receive request status values 3–11
3–6 Status value meanings 3–14
4–1 LAN Block 4–1
4–2 LDC 4–2
4–3 VCIB structure 4–4
4–4 VCRP structure 4–9
4–5 DCBE structure 4–15
4–6 LIL header structure 4–17
4–7 LIL item structure 4–18
6–1 Transmit request fields 6–7
6–2 Receive complete fields 6–8
A–1 Registered VCI users A–1
B–1 Revision history B–1

Chapter 1

INTRODUCTION

The VMS LAN drivers have an unprivileged interface (QIO) that has been available since the introduction of the LAN drivers. This specification describes a privileged interface to the same VMS LAN drivers. In some sections, the QIO interface is referenced. The QIO interface to the LAN drivers is documented in the LAN chapter of the VMS I/O User's Manual.

The privileged interface to the VMS LAN drivers is the VMS Communications Interface (VCI). It is an interface that is available at IPL 8 that has been designed to be very efficient. For these reasons, the VCI is the preferred interface for those applications that require minimal CPU overhead and/or maximum LAN utilization without the protection that the QIO interface provides.

This document assumes that the reader is familiar with the internals of VMS. The following is a list of documents which are relevant to this document:

- VMS I/O User's Manual (LAN Chapter)
- VMS Internals and Data Structures

1.1 CONTENTS

This document contains a description of VCI terms, data structures, and routines that form the LAN VCI.

1.2 PERFORMANCE

Compared to QIO, this interface uses less CPU and less memory for the transmission and reception of data on the LAN. Within the LAN drivers, the VCI code path is optimized over the QIO code path.

1.3 SUPPORT OBJECTIVES

The requirements and recommendations of the LAN VCI are subject to change between releases (major or minor) of VMS. As optimizations to this interface are developed they will be incorporated into the interface. Implementations using this interface must be updated to reflect any such changes.

The LAN VCI is not supported on the DEQNA device. Attempts to use the LAN VCI on a DEQNA result in the error SS$_UNSUPPORTED being returned.

This specification assumes that MACRO-32 will be the programming language used. However, the use of a high level language is not precluded.

The LAN VCI on OpenVMS VAX and OpenVMS AXP are mostly the same. This document describes the LAN VCI available in OpenVMS VAX version 6.0 and OpenVMS AXP version 1.5. There are a few data structure fields in OpenVMS AXP version 1.5 that have been extended to a longword (4 bytes). In a future release of OpenVMS VAX, the same fields will be extended to match the data structures in OpenVMS AXP version 1.5.

Differences in the LAN VCI of OpenVMS VAX version 6.0 and OpenVMS AXP version 1.5 are described in this specification.

Chapter 2

VCI OVERVIEW

The LAN VCI is an interface that allows the creation of ports to a LAN driver, allows the port to be enabled with specific attributes, allows the port to be used for data transmission and reception, allows the port to be disabled, and allows for the deletion of the port. The functions available from the LAN VCI are very similar to the functions available from the LAN QIO interface.

The LAN VCI has the following characteristics:

1. The LAN VCI uses a JSB/RSB calling mechanism.
2. The LAN VCI defines requests that have no need for process context.
3. The LAN VCI allows for the transmission of data directly from process space or system space.
4. The LAN VCI runs at IPL 8 with the IOLOCK8 spinlock.

The rest of this chapter provides a definition of terms, an overview of the primary data structures, and an overview of the VCI routines. The details are provided in following chapters.

2.1 TERMS

This section describes the basic terms used within this specification.

2.1.1 VMS COMMUNICATION MODULE

A VMS Communication Module (VCM) is a single piece of VMS kernel mode code which implements a protocol and provides a VCI and/or uses a VCI. The LAN drivers are VCMs and each user of the VCI to the LAN drivers is a VCM. It is not required that a single VCM map directly to a single image or that a single image represent only one VCM. However, it is typical for a VCM to be implemented as a single image. Another example of a VCM is LTDRIVER (the VMS pseudo driver that implements the LAT protocol).

In keeping with architecture layering, an individual VCM has a hierarchical relationship to its adjacent VCM. Therefore, from a given VCM there may be an upper VCM and/or a lower VCM. The upper VCM implements the protocol of the adjacent higher layer of the architecture. The lower VCM implements the adjacent lower layer of the architecture. With the LAN VCI, the LAN driver is always the lower VCM and the user of the LAN VCI is the upper VCM.

2.1.2 SERVICE ROUTINE

The VCI routines are divided into several services. Each service provides a different type of request. For example, there is a port management service and there is a transmit service. These two services provide different functions. Each service may have one or more service routines. The service routines form the interface between the two adjacent VCMs. Each service routine exists in either the upper VCM or the LAN driver. A service with multiple service routines can have some of its service routines exist in the upper VCM and the rest in the LAN driver.

In general, the LAN driver provides INITIATE service routines and the upper VCM provides COMPLETION service routines. In some situations, the completion or initiation routine is not needed because the completion of the request or the initiation of the request is implied.

An example of a service that has both initiate and completion service routines is the TRANSMIT service. This service has an INITIATE routine which resides in the LAN driver and a COMPLETION routine which resides in the upper VCM.

An example of a service that has only a completion service routine is the REPORT EVENT service where there is a report event routine in the upper VCM only.

An overview of the service routines is available in Section 2.3 and a detailed description of each service routine is available in Chapter 3.

2.1.3 VCI PORT

The VCI port can be considered a "channel" that's established between two VCMs using the VCI. It comprises the handle for the communication between the two VCMs. The VCI port is represented by a VCIB data structure (described in Section 2.2.3). Once a VCIB has been initialized by both the upper VCM and the LAN driver, a VCI port has been created. Note that two VCMs may have more than one VCI port between them; and for each VCI port, there is a unique VCIB.

Although the VCI port is somewhat equivalent to a QIO channel, it cannot be accessed through QIO nor through the $ASSIGN system service. For each VCI port, the LAN driver creates a UCB; just as the I/O subsystem creates a UCB for each QIO channel. The UCB that's associated with the VCI port is visible using SDA; but is NOT available for access by the upper VCM.

Note that the VCI port is a VCI concept and is not associated with an architectural port or any other entity called a port.

2.2 OVERVIEW OF DATA STRUCTURES

There are five major data structures used in the LAN VCI. The data structures are described in full detail in Chapter 4.

2.2.1 LAN BLOCK

The LAN block is a data structure that's allocated and owned by the LAN drivers. The address of the LAN block is stored in the system data cell NET$AR_LAN_VECTOR. The LAN block contains some information that can be used by the upper VCM and it contains addresses of some routines that the upper VCM calls.

2.2.2 LAN DEVICE CHARACTERISTICS

The LAN Device Characteristics (LDC) data structure is owned by the LAN drivers. There is one LDC for each available LAN device. The upper VCM can acquire the address of each LDC by calling an access routine repeatedly. The LDC can be used by the upper VCM to determine which available LAN devices to use.

2.2.3 VCI BLOCK

The VCI Block (VCIB) is the data structure used to represent a VCI Port. It contains the information required for the two VCMs to keep context on the VCI port and for the two VCMs to call each other. The upper VCM is responsible for allocating the VCIB, initializing its fields in the VCIB, and calling the LAN driver to initialize its fields in the VCIB. The VCIB contains pointers to the services provided by the upper VCM and the LAN driver.

2.2.4 VMS COMMUNICATIONS REQUEST PACKET

The VMS Communications Request Packet (VCRP) is used to exchange information and completion status of requests between VCMs. It describes the type of function to be performed and provides the parameter information needed for the request. The VCRP is allocated and initialized by a VCM. It is then passed to the upper VCM or the LAN driver at the appropriate service address.

2.2.5 DATA CHAIN BLOCK

The Data Chain Block (DCBE) is used to connect a buffer containing all or part of a packet to a VCRP. This allows a packet to exist in multiple non-contiguous buffers of memory. When a packet is broken up into multiple buffers, it is called chaining. Chaining can occur on transmit requests only.

2.3 OVERVIEW OF ROUTINES

This section contains an overview of the services that make up the LAN VCI. The section is divided into five parts.

1. Access Routines
2. Port Management Service
3. Transmit Service
4. Receive Service

5. Report Event Service

Each classification of services is made up of a set of service routines that support the classification. Each of the following sections describe the service routines of a specific service classification. All the service routines are described in full detail in Chapter 3.

2.3.1 ACCESS ROUTINES

The access routines are provided to allow the upper VCM to learn which LAN devices are available and to create and delete VCI ports to those LAN devices.

The Access routines include:

- LAN$GET_DEVICE (in the LAN driver) is used to determine which LAN devices are available on the system by returning the address of an LDC data structure.
- VCI$LAN_CREATE_PORT (in the LAN driver) is used to create a VCI port between the upper VCM and the LAN driver.
- VCI$LAN_DELETE_PORT (in the LAN driver) is used to delete a VCI port.

2.3.2 PORT MANAGEMENT SERVICE

Ports represent the connection or channel between two VCMs. The Port Management services are provided to allow the upper VCM to enable (start), change, and disable (shut down) VCI ports to the LAN drivers.

The Port Management services include:

- VCI$LAN_PORTMGMT_INITIATE (in the LAN driver) is used to enable (start), change, and disable (shut down) a VCI port.
- VCI$xxx_PORTMGMT_COMPLETE (in the upper VCM) is used to complete all requests passed to the VCI$LAN_PORTMGMT_INITIATE routine.

2.3.3 TRANSMIT SERVICE

The transmit service is used to support the transmission of packets. In this service, the upper VCM constructs a transmit VCRP and initiates the request using the address of the service routine provided by the LAN driver.

The Transmit services include:

- VCI$LAN_TRANSMIT_INITIATE (in the LAN driver) is called to initiate a transmit request in the LAN driver. The LAN driver builds the LAN header for this packet in the VCRP and then transmits the resulting frame.
- VCI$LAN_BUILD_HDR (in the LAN driver) is called to have the LAN driver build a LAN header for the upper VCM.
- VCI$LAN_TRANSMIT_FRAME (in the LAN driver) is called to initiate a transmit request in the LAN driver. The upper VCM will have already included a pre-built LAN header into the packet to be transmitted.

- VCI$xxx_TRANSMIT_COMPLETE (in the upper VCM) is called to complete a transmit request and return its status to the upper VCM.

2.3.4 RECEIVE SERVICE

The receive service is used to support the reception of packets. In this service, the upper VCM does not need to post read requests. Instead, the LAN driver calls the upper VCM's receive completion routine with packets received for that VCM.

The Receive services include:

- VCI$xxx_RECEIVE_COMPLETE (in the upper VCM) is called to pass a receive packet to the upper VCM.

2.3.5 REPORT EVENT SERVICE

When an asynchronous event is detected by the LAN driver, it may wish to signal the event to the upper VCM. The Report Event service is used to perform this notification.

The Report Event services include:

- VCI$xxx_REPORT_EVENT (in the upper VCM) is used to report an asynchronous event to the upper VCM.

Chapter 3

ROUTINE DESCRIPTIONS

This chapter describes the rules for calling the routines of the VCI.

Each update to the VCI is assigned a new version number. A new release of VMS may contain a new (higher) version of the VCI or it may contain the same version as the previous release of VMS.

All routines are called according to the following synchronization rules. All routines are called at IPL 8 with IOLOCK8 as the highest lock acquired. On a system where SMP is not enabled, no locks need be acquired. All routines are called on a CPU where access to I/O space is allowed - typically this means that the fork thread must be running on the primary CPU. The LAN drivers call the upper VCM according to these rules. The upper VCMs must adhere to these rules also.

See Chapter 6 for detailed steps in using these routines.

3.1 ACCESS ROUTINES

This section describes the Access Routines. These routines are accessed through the LAN block.

3.1.1 LAN$GET_DEVICE

For each LAN device, some amount of information about the LAN device is made available to the upper VCM.

This routine is typically called repeatedly so that information about all devices can be acquired by the upper VCM. Each call to this routine provides the upper VCM with information about the next available LAN device.

Each call to this routine returns a pointer to the LDC structure of an existing LAN device or zero if there are no more existing LAN devices.

The call to LAN$GET_DEVICE is made through the LAN block. The calling sequence is as follows:

```
MOVL    G^NET$AR_LAN_VECTOR,R0
BEQL    NO_LAN_BLOCK
CMPL    #1,LAN$L_VERSION(R0)
BNEQ    WRONG_VERSION
JSB     @LAN$A_GET_DEVICE(R0)
```

INPUTS:

R0 = Last ID returned from this routine; or zero if this is the first call

OUTPUTS:

R0 = ID for this device - this ID can be passed back to this routine in a subsequent call to get the next device. However, the caller should keep all synchronization in place between calls to this routine.

R1 = either the address of the LDC for a device or the value zero if there are no more devices.

This call is completed synchronously.

The upper VCM can use this routine to get the name of the first LAN device by calling the routine once. The upper VCM could also use this routine to determine the name of the first FDDI device by calling this routine repeatedly until an FDDI device is returned or until a 0 is returned. The upper VCM would look in each LDC to see if that LDC was for an FDDI device. The upper VCM could also use this routine to determine the name of all the LAN devices.

Once using this routine to acquire the name of the desired LAN device, the upper VCM would typically pass the name of the device to the VCI$LAN_CREATE_PORT routine to create a VCI port to that LAN device.

If the upper VCM already knows the name of the desired device, then the LAN$GET_DEVICE routine need not be called.

3.1.2 VCI$LAN_CREATE_PORT

The create port service is provided by the LAN driver. The address of the service is stored in the LAN block. The service is used to create a VCI Port.

Upon successful return from the create port routine, the VCI Port is in a state where it can be used to issue VCRPs between the VCMs. A successful call to the create port routine implies the creation of a VCI port; it does not imply that the port is now able to be used for packet transmission or reception. The upper VCM has to enable the port and wait for the port usable event before packet transmission or reception can begin.

Note that the LAN drivers cannot deallocate the VCIB; since that data structure is owned by the upper VCM. And the upper VCM cannot deallocate the VCIB until the VCI port has been deleted by a successful call to the delete port routine.

The steps to using the Create Port service are:

1. The upper VCM makes sure that the LAN block exists, verifies the version of the LAN block, and verifies that the address of the CREATE_PORT routine has been stored in the LAN block by the LAN drivers.

2. The upper VCM allocates the VCIB and initializes its portion of the VCIB.
3. The upper VCM calls the LAN driver's create port routine to allow the LAN driver to initialize its portion of the VCIB. Remember that this is a synchronous call - it is completed when control is returned to the upper VCM.
4. Upon successful completion of the call to the LAN driver's create port routine, the upper VCM may be required to check the version number in the VCIB to ensure that the LAN driver VCI version is compatible with the version of the upper VCM. To do this, the upper VCM first checks to see if its version is higher than the LAN driver's version. If not, then the upper VCM does not have to perform the check. If it is higher, then the upper VCM has to perform the check. At this point, the only supported version is the number 1. If the upper VCM finds that the LAN driver's version is lower than 1, then the upper VCM should delete the VCI port and not use that device.

The call to create the VCI Port is made through the LAN block. The calling sequence is as follows:

```
MOVL    G^NET$AR_LAN_VECTOR,R0
BEQL    NO_LAN_BLOCK
CMPL    #1,LAN$L_VERSION(R0)
BNEQ    WRONG_VERSION
JSB     @LAN$A_CREATE_PORT(R0)
```

INPUTS:

R4 = Address of VCIB to initialize

OUTPUTS:

R0 = Completion status

This call is completed synchronously.

The status of the call is stored in R0 before returning to the caller. Table 3–1 shows some of the possible values. See Section 3.6 for a meaning of the status values. Only the first status code is a success status value; the others are failure status values. Other error status values may be returned.

Table 3–1: Create port status values

R0
SS$_NORMAL
SS$_DEVACTIVE
SS$_INSFMEM
SS$_NOSUCHDEV
SS$_SYSVERDIF
SS$_UNSUPPORTED

3.1.3 VCI$LAN_DELETE_PORT

The delete port service is provided by the LAN driver. The address of the service is stored in the LAN block. The service is used to delete a VCI Port.

Upon successful return from the delete port routine, the VCI Port no longer exists. At this point, the upper VCM can deallocate the VCIB and any other context it has for the VCI port. The LAN drivers do NOT deallocate the VCIB.

The steps to using the Delete Port service are:

1. The upper VCM calls the LAN driver's delete port routine to allow the LAN driver to clean up its context on the VCI port. Remember that this is a synchronous call - it is completed when control is returned to the upper VCM.
2. Upon successful completion of the call to the LAN driver's delete port routine, the upper VCM may deallocate the VCIB and clean up any other context it has on the VCI port.

The call to delete the VCI Port is made through the LAN block. The calling sequence is as follows:

```
MOVL    G^NET$AR_LAN_VECTOR,R0
JSB     @LAN$A_DELETE_PORT(R0)
```

INPUTS:

R4 = Address of VCIB of port to delete

OUTPUTS:

R0 = Completion status

This call is completed synchronously.

The status of the call is stored in R0 before returning to the caller. Table 3–2 shows some of the possible values. See Section 3.6 for a meaning of the status values. Only the first status code is a success status value; the others are failure status values. Other error status values may be returned.

Table 3–2: Delete port status values

R0
SS$_NORMAL
SS$_DEVACTIVE
SS$_NOSUCHDEV

3.2 PORT MANAGEMENT SERVICE ROUTINES

This section describes the Port Management Service Routines. These routines are accessed through the VCIB.

3.2.1 VCI$LAN_PORTMGMT_INITIATE

The PORTMGMT_INITIATE service is provided by the LAN drivers.

PORTMGMT_INITIATE requests allow the upper VCM to perform functions on the VCI port. Note that once the VCI port has been created, it is still not totally functional. For example, it may not be used to transmit and receive packets. Using PORTMGMT_INITIATE requests, the VCI port can be enabled (started) with specific attributes, changed, or disabled (shut down).

Since the PORTMGMT_INITIATE service allows different operations, a function code in the VCRP is used to distinguish from the different types of PORTMGMT_INITIATE requests. The function codes that may be used with the PORTMGMT_INITIATE service routine are listed below. Some of these function codes are defined in $VCRPDEF in LIB.MLB and some are defined in $VCRPLANDEF in LANUDEF.MLB.

- The VCRP$K_FC_ENABLE_PORT function allows the upper VCM to set the port's attributes and put the port into a state where packets can be transmitted and received.
- The VCRP$K_FC_LAN_CHANGE_PORT function allows the changing of values of certain attributes of the VCI port.
- The VCRP$K_FC_DISABLE_PORT function allows the VCM to set the port's state to off; where packets cannot be transmitted and received.

This service completes its requests asynchronously. That is, requests are completed through another service routine (the PORTMGMT_COMPLETE service routine).

Since requests are completed through the VCI$xxx_PORTMGMT_COMPLETE service routine, there is no return status on the call to the VCI$LAN_PORTMGMT_INITIATE routine. The status of the request is stored in the VCRP$Q_REQUEST_STATUS field of the VCRP when the VCRP is returned to the upper VCM.

The calling sequence is as follows:

```
JSB     @VCIB$A_PORTMGMT_INITIATE(R4)
```

INPUTS:

R3 = Address of VCRP
R4 = Address of VCIB

OUTPUTS:

R0 and R1 are destroyed

This call is completed asynchronously through the upper VCM's Port Management Complete service.

3.2.2 VCI$xxx_PORTMGMT_COMPLETE

When the LAN driver has completed processing a request made through the VCI$LAN_PORTMGMT_INITIATE service, it calls the VCI$xxx_PORTMGMT_COMPLETE service in the upper VCM to return the VCRP.

The calling sequence is as follows:

```
JSB     @VCIB$A_PORTMGMT_COMPLETE(R4)
```

INPUTS:

R3 = Address of VCRP
R4 = Address of VCIB

OUTPUTS:

R0 and R1 are destroyed

The status of the request is stored in the VCRP$Q_REQUEST_STATUS field of the VCRP. Table 3–3 shows some of the possible values for this quadword field. See Section 3.6 for a meaning of the status values in the first word. Only the first status code is a success status value; the others are failure status values. Other error status values may be returned.

Table 3–3: Port management request status values

1st word	2nd word	3rd word	4th word
SS$_NORMAL	0	0	0
SS$_ABORT	0	0	0
SS$_BADPARAM	bad parameter	0	0
SS$_DEVACTIVE	0	0	0
SS$_DEVINACT	0	0	0
SS$_DEVOFFLINE	0	0	0
SS$_ILLIOFUNC	0	0	0
SS$_INSFMEM	0	0	0
SS$_IVADDR	0	0	0
SS$_OPINCOMPL	0	0	0
SS$_TIMEOUT	0	0	0

3.3 TRANSMIT SERVICE ROUTINES

The transmit service is used to support the transmission of packets. In this service, the upper VCM constructs a transmit VCRP and initiates the request using the address of one of the two service routines provided by the LAN driver. The transmit request is asynchronous. This means that the LAN driver completes the transmit request via the transmit completion service routine of the upper VCM.

If the LAN driver cannot accept the request (possibly it has not been enabled), then the transmit is completed immediately via the upper VCM's transmit completion service routine.

IMPLEMENTOR'S NOTE

The transmit error case can cause problems. The two VCMs could call recursively with the upper VCM requesting transmits and the LAN driver returning an error on every transmit request. This could cause a system hang at a high IPL possibly with an important spinlock. Or the recursive calls between the two VCMs could overflow the stack and crash the system. It is left to the upper VCM to handle this situation.

Transmit requests may be issued to the LAN driver after the upper VCM has received a PORT_USABLE event. The PORT_USABLE event is reported to the upper VCM after a successful completion of an ENABLE_PORT request and the device is operational. When the upper VCM receives a PORT_UNUSABLE event, the upper VCM is not allowed to make transmit requests until it receives another PORT_USABLE event.

Most LAN Drivers are able to handle chained and non-chained transmit requests. The rules for making chained and non-chained transmit requests are described in Section 6.4.2. If the LAN driver does not support chained transmit requests, then the VCIB$V_DLL_XMT_CHAIN flag is set in the field VCIB$W_DLL_FLAGS and the upper VCM should not make chained requests. On OpenVMS AXP, transmit chaining will always be supported, so the VCIB$V_ DLL_XMT_CHAIN flag will never be set. The upper VCM does not need to check this flag on OpenVMS AXP.

The VCRP$L_BOFF field must be initialized to allow sufficient space for the LAN driver to back-build its header in the VCRP without going into the scratch field of the VCRP. Note that even though it's required that the LAN driver not back-build its header into the scratch space of the VCRP, the LAN driver doesn't check that this occurs. This is because the VCI is a privileged interface and we expect the upper VCM to initialize the VCRP properly. Therefore the upper VCM has to initialize VCRP$L_BOFF such that it allows enough space for the LAN header. The field VCIB$W_DLL_HDR_SIZE can be used by the upper VCM to ensure that it leaves enough space for the LAN header. The upper VCM can also use the constant value LAN$C_MAX_HDR_SIZE as the maximum LAN header size. If LAN$C_MAX_HDR_SIZE is not defined in $LANUDEF in LANUDEF.MLB, then the constant 58 (decimal) can be used.

There are two ways to make a transmit request to the LAN drivers:

1. The upper VCM calls the transmit initiate routine to transmit data. The LAN driver builds its header in the VCRP and passes the packet to the device. When the device completes the request to the LAN driver, the LAN driver calls the transmit complete routine in the upper VCM. This style of transmit request is very similar to the QIO request (except that the overhead in CPU time is greatly reduced as compared to QIO). The upper VCM provides the standard information required by the LAN driver to build the header.

2. A more efficient method to transmit packets is for the upper VCM to call the build header routine in the LAN driver. The upper VCM then places this header in all (or most) of the VCRPs it uses for transmit requests and calls the LAN driver's transmit frame routine. This style of transmit requests allows for the LAN header to be built once. The CPU time required to build the header for every packet is saved if the same header is used for multiple transmit requests.

The status for a transmit request is stored in the VCRP$Q_REQUEST_STATUS field of the VCRP when the VCRP is returned to the upper VCM.

The transmit service routines are called through the VCIB. They are described in the following sections.

3.3.1 VCI$LAN_TRANSMIT_INITIATE

Transmit requests with no LAN header are made to the LAN driver via a call to the VCI$LAN_TRANSMIT_INITIATE routine. Completion of the VCRP is done via the transmit complete service.

The calling sequence is as follows:

```
JSB     @VCIB$A_TRANSMIT_INITIATE(R4)
```

INPUTS:

R3 = Address of VCRP
R4 = Address of VCIB

OUTPUTS:

R0, R1, and R3 are destroyed

3.3.2 VCI$LAN_BUILD_HDR

To request to have a LAN header built, the upper VCM calls the LAN driver's VCI$LAN_BUILD_HDR routine. The upper VCM passes some header information to this routine. Completion of this request is made synchronously.

The calling sequence is as follows:

```
JSB     @VCIB$A_LAN_BUILD_HDR(R4)
```

INPUTS:

R0 = Address of where the header should be backfilled from

R1 = If this is an 802 format port, then R1 should point to the 802 transmit longword (see the definition of the VCRP$L_LAN_802XMT field in Section 4.4). To use defaults, the upper VCM has R1 point to a longword containing the value zero.

R2 = If this is an 802 format port, then R2 should point to the byte containing the response flag.

R4 = Address of VCIB

OUTPUTS:

R0 = Address of beginning of header
R3 = Size of header

If this is an 802 format port and R1 pointed to a value of zero, then the longword pointed to by R1 and the byte pointed to by R2 are modified to contain the default values.

3.3.3 VCI$LAN_TRANSMIT_FRAME

Transmit requests with a LAN header are made to the LAN driver via a call to the VCI$LAN_TRANSMIT_FRAME routine. Completion of the VCRP is done via the transmit complete service.

The calling sequence is as follows:

```
JSB     @VCIB$A_LAN_TRANSMIT_FRAME(R4)
```

INPUTS:

R3 = Address of VCRP
R4 = Address of VCIB

OUTPUTS:

R0, R1, and R3 are destroyed

3.3.4 VCI$xxx_TRANSMIT_COMPLETE

When the LAN driver has completed processing a transmit request, it calls this service in the upper VCM to return the transmit VCRP.

The calling sequence is as follows:

```
JSB     @VCIB$A_TRANSMIT_COMPLETE(R4)
```

INPUTS:

R3 = Address of VCRP
R4 = Address of VCIB

OUTPUTS:

R0, R1, and R3 are destroyed

The status of the request is stored in the VCRP$Q_REQUEST_STATUS field of the VCRP. Table 3–4 shows some of the possible values for this quadword field. See Section 3.6 for a meaning of the status values in the first word. Only the first status code is a success status value; the others are failure status values. Other error status values may be returned.

Table 3–4: Transmit request status values

1st word	2nd word	3rd word	4th word
SS$_NORMAL	undefined	undefined	0
SS$_ABORT	0	0	0
SS$_DEVINACT	0	0	0
SS$_IVBUFLEN	0	0	0
SS$_OPINCOMPL	0	0	0
SS$_THIRDPARTY	0	0	0
SS$_TIMEOUT	0	0	0

3.4 RECEIVE SERVICE ROUTINES

The receive service is used to support the reception of packets. The receive service routines are called through the VCIB.

The LAN drivers use an "allocate receive" method for handling receive packets. This involves the LAN driver allocating receive buffers on behalf of the upper VCM. The upper VCM never needs to allocate a receive buffer and never needs to post read requests.

In the allocate receive method the LAN driver allocates the receive buffers. The LAN driver supplies no service for initiating receives. The initiation of receives is implicit when the VCI port becomes usable. The upper VCM supplies the receive complete service routine. Receive buffers are passed to the upper VCM through the receive complete service routine.

An important point about the allocate receive method is that since there is no initiation of the request, the receive VCRP does not traverse the VCMs from top to bottom and then bottom to top. The receive VCRP begins in the LAN driver and traverses the VCMs once from the bottom to the top.

There are different ways for the upper VCM to process the receive VCRPs. The upper VCM can either hold onto the VCRP or it can return the VCRP before it returns control back to the LAN driver. When the upper VCM decides it's done with the VCRP, it returns the VCRP to the LAN driver by calling the routine pointed to by the VCRP$A_DEALLOC_RTN field in the receive VCRP. This is the only way an upper VCM can get rid of a receive VCRP.

3.4.1 VCI$xxx_RECEIVE_COMPLETE

When the LAN driver has received a packet for a usable port of an upper VCM, it calls this service in the upper VCM to pass the VCRP and the packet.

The calling sequence is as follows:

```
JSB     @VCIB$A_RECEIVE_COMPLETE(R4)
```

INPUTS:

R3 = Address of VCRP

R4 = Address of VCIB

OUTPUTS:

R0, R1, and R3 may be destroyed

The status of the request is stored in the VCRP$Q_REQUEST_STATUS field of the VCRP. Table 3–5 shows the possible values for this quadword field. See Section 3.6 for a meaning of the status values in the first word.

Table 3–5: Receive request status values

1st word	2nd word	3rd word	4th word
SS$_NORMAL	undefined	0	0

3.5 REPORT EVENT SERVICE ROUTINES

The LAN driver reports the following events to the upper VCM through the report event service routine supplied by the upper VCM. The report event service routines are called through the VCIB.

The calling sequence is as follows:

```
JSB     @VCIB$A_REPORT_EVENT(R4)
```

INPUTS:

R1 = Event code
R2 = Reason code (optional)
R4 = Address of VCIB

OUTPUTS:

R0 and R1 may be destroyed

The event codes are defined in $VCRPLANDEF in LANUDEF.MLB.

3.5.1 PORT USABLE EVENT

When this event is reported it signals the upper VCM that its port can be used for data transfer.

The LAN driver reports this event whenever the VCI port becomes ready for data transfer. This can happen due to an ENABLE_PORT request from the upper VCM or due to the completion of an automatic restart following an error that made the port unusable.

The input registers for this event are:

- R1 = Event code = VCRP$K_EC_DLL_PORT_USABLE

- R4 = VCIB address

3.5.2 PORT UNUSABLE EVENT

When this event is reported it signals the upper VCM that the VCI port can no longer be used for data transfer. The port can become unusable for many reasons. For example, it can be disabled by a DISABLE_PORT request and whenever the LAN driver has a fatal error. The reasons that the port became unusable are described below as part of the reason codes for this event.

The port can be re-enabled by the upper VCM (using an ENABLE_PORT request) or automatically by the LAN driver if the upper VCM has enabled this feature.

This event and the PORT_USABLE event are reported alternately. That is, the upper VCM never receives back to back PORT_USABLE events and never receives back to back PORT_UNUSABLE events.

The reason codes passed to the upper VCM for this event are:

- SS$_NORMAL - Disable port requested by upper VCM
- SS$_THIRDPARTY - A data link entity was disabled by management action
- SS$_SSFAIL - The LAN driver detected an error and has reset the device

The input registers for this event are:

- R1 = Event code = VCRP$K_EC_DLL_PORT_UNUSABLE
- R2 = Reason code
- R4 = VCIB address

3.5.3 RECEIVE CONGESTION EVENT

The LAN driver reports this event whenever receive packets are lost (whenever the Station Buffer Unavailable counter is incremented). This event will not be reported any more than once per five minutes. This is to prevent excessive overhead when we should be trying to handle the incoming packets.

The input registers for this event are:

- R1 = Event code = VCRP$K_EC_LAN_RCV_CONGESTION
- R4 = VCIB address

3.5.4 NEW ADDRESS EVENT

The LAN driver reports this event when the physical address of the station has been changed.

The input registers for this event are:

- R1 = Event code = VCRP$K_EC_LAN_NEW_ADDRESS
- R2 = Memory address of new physical address
- R4 = VCIB address

Note that when the LAN driver reports the new address event, all headers supplied to the upper VCM through the BUILD_HDR service routine are invalid. The upper VCM has to re-issue all BUILD_HDR requests and replace all the old headers with new headers. The size of the headers remain the same; just the contents of the header will have changed.

3.5.5 RECEIVE PDU LOST EVENT

The LAN driver reports this event when:

1. the upper VCM has put a 1 in the field VCIB$V_LAN_RCV_LIM and
2. the field VCIB$W_LAN_MAX_RCV is equal to the field VCIB$W_LAN_OUT_RCV and
3. another frame has been received by the LAN driver for this upper VCM.

The frame received for the upper VCM is discarded by the LAN driver.

The input registers for this event are:

- R1 = Event code = VCRP$K_EC_LAN_RCV_PDU_LOST
- R4 = VCIB address

3.5.6 RESTART FAILED EVENT

The LAN driver reports this event when it has exhausted its attempts to automatically restart the port. When this event is reported, the upper VCM should delete the port or attempt to enable the port.

The input registers for this event are:

- R1 = Event code = VCRP$K_EC_LAN_RESTART_FAIL
- R4 = VCIB address

3.5.7 STATION RENAMED EVENT

The LAN driver reports this event when it cannot restart the upper VCM because the DECnet Phase V customer assigned name of the station has been changed. This can happen if network management disables and deletes the station and then creates and enables the same station; but with a different name. When this event is reported, the upper VCM must delete the port.

The input registers for this event are:

- R1 = Event code = VCRP$K_EC_LAN_STATION_RENAMED
- R4 = VCIB address

3.6 STATUS VALUES

This section describes some of the different values that can be returned to the upper VCM in the VCRP request status field and in R0.

Table 3–6: Status value meanings

Status value	Definition
SS$_NORMAL	The request has completed successfully.
SS$_ABORT	All VCI ports are being disabled due to an error detected by the LAN driver.
SS$_BADPARAM	One of the parameters in the request was not specified correctly. For ENABLE_ PORT, the NMA$C_PCLI_xxx code of the P2 parameter in error is in the second word of VCRP$Q_REQUEST_STATUS.
SS$_DEVACTIVE	The VCI Port is active.
SS$_DEVINACT	The VCI Port is not active.
SS$_DEVOFFLINE	The LAN device is OFFLINE and is not usable.
SS$_ILLIOFUNC	The function code field of the VCRP had a value that did not represent a function known by the LAN driver.
SS$_INSFMEM	There was not enough non-paged pool to complete the request.
SS$_IVADDR	The upper VCM's PHA parameter is the same as another node on the extended LAN.
SS$_IVBUFLEN	The number of bytes to transmit was too large.
SS$_NOSUCHDEV	The device name passed does match an existing device.
SS$_OPINCOMPL	The request could not be completed because the VCI port is being restarted.
SS$_SYSVERDIF	The LAN driver's VCI version was higher than the upper VCM's version and they are not compatible versions.
SS$_THIRDPARTY	A data link entity was disabled by management action.
SS$_TIMEOUT	The VCI port is being disabled due to a timeout.
SS$_UNSUPPORTED	The LAN driver does not support the VCI and/or the VCM (by VCI ID) on this device.

Chapter 4

DATA STRUCTURES

This section describes the data structures used by the VCI.

For efficiency, the LAN drivers require that the VCIB, VCRP, and DCBE be quadword aligned. The upper VCM must ensure that these data structures are quadword aligned.

All these data structures must exist in system space.

4.1 LAN BLOCK

The LAN block is a LAN driver data structure that is used by the upper VCM for calling the access routines. The fields in this data structure begin with LAN$.

The LAN block fields are defined by $LANUDEF in LANUDEF.MLB.

Table 4–1 describes the fields of the LAN block.

Table 4–1: LAN Block

FIELD NAME	ACCESS BY UPPER VCM	ACCESS BY LAN DRIVER	DESCRIPTION
L_VERSION	R	W	LAN VCI version
A_GET_DEVICE	R	W	Address of Get Device routine
A_CREATE_PORT	R	W	Address of Create Port routine
A_DELETE_PORT	R	W	Address of Delete Port routine

Key to Access

W: The module has read and write access to the field.
R: The module has read access to the field.

Note that although the LAN block may exist, these fields may not be initialized. These fields are initialized when the LAN$L_VERSION field is non-zero. A detailed description of each of these LAN block fields follows:

- LAN$L_VERSION

  This field is used by the upper VCM to check that the rest of the LAN block fields have been initialized and that the version of the LAN block is compatible with their software. The upper VCM must check that this field contains the value 1 before using the other fields of the LAN block.
- LAN$A_GET_DEVICE

  The LAN driver stores the address of the get device routine in this field.
- LAN$A_CREATE_PORT

  The LAN driver stores the address of the create port routine in this field.
- LAN$A_DELETE_PORT

  The LAN driver stores the address of the delete port routine in this field.

4.2 LDC

The LDC is a LAN driver data structure that is used by the upper VCM to acquire information about a LAN device. The fields in this data structure begin with LDC$. The LDC data structure is quadword aligned in memory.

The LDC fields are defined by $LDCDEF in LANUDEF.MLB.

Table 4–2 describes the fields of the LDC.

Table 4–2: LDC

FIELD NAME	ACCESS BY UPPER VCM	ACCESS BY LAN DRIVER	DESCRIPTION
A_NAME	R	W	Address of device name string
(W)L_TYPE	R	W	Type of medium
(W)L_RCVSIZE	R	W	Minimum number receive ring entries
(B)L_DEVTYPE	R	W	VMS device type

Field name

If the field name begins with a letter in parentheses, then the letter in the parentheses is used in the OpenVMS VAX name.

Key to Access

W: The module has read and write access to the field.
R: The module has read access to the field.

Remember that the address of the LDC is returned on calls to the LAN$GET_DEVICE routine. The address of the LDC is also stored (by the LAN driver on a call to VCI$LAN_CREATE_PORT) in the VCIB if the upper VCM decides to use the device; so the upper VCM does not need to save the LDC address in its own data structure. A detailed description of each of these LDC fields follows:

- LDC$A_NAME

  This field contains the address of a counted string containing the device name. Note that this counted string is exactly what is required to be passed to the CREATE_PORT routine in the DLL$K_LAN_DEVICE item of the item list pointed to by VCIB$A_DLL_INPUT_LIST. DLL$K_LAN_DEVICE is defined in $LANUDEF in LANUDEF.MLB.

- LDC$W_TYPE

  This field is used by the upper VCM to detect the type of medium for this LAN device. The value stored in this field is the same as the value stored in the field VCIB$W_DLL_TYPE. See Section 4.3 for the list of values for that field.

- LDC$W_RCVSIZE

  This field contains the minimum number of receive buffers that are given to this device when there is sufficient non-paged pool and CPU time to fill the device's receive buffer pool. If there is insufficient non-paged pool or if there isn't enough CPU time to keep the device's receive buffer pool full, then the actual number of receive buffers owned by the device may be less than the value stored in this field. The LAN driver is usually able to keep the number of receive buffers given to the device at or above the number stored in this field. The upper VCM can use this number to inform the other nodes on the network how many packets to send to this node. Note that no upper VCM should assume that it can have all these receive buffers available at all times. The LAN is a shared medium. It is suggested that no more than 75% of the receive buffers be used as a pipeline.

- LDC$B_DEVTYPE

  This field contains the VMS device type for this device. This is the same as the value stored in the UCB$B_DEVTYPE field for this device. The device types are listed in the LAN chapter of the VMS I/O User's Manual and are defined in $DCDEF in STARLET.MLB.

4.3 VCIB

The VCI block is allocated, owned, and deallocated by the upper VCM. Neither VCM is allowed to deallocate a VCI block while the VCI Port for that VCIB is created. The LAN driver never deallocates the VCIB of an upper VCM.

Note that there are fields in the VCIB that are initialized, used, and referenced only by the LAN driver. The size of the VCIB as defined by this specification is the size required by the LAN drivers. The upper VCM is allowed to make the VCIB larger and define its own VCIB fields in the portion of the VCIB following the size needed by the LAN drivers. It is expected that most upper VCMs will do this to allow for context required by the upper VCM.

The registered name for this data structure is VCIB$.

The VCIB fields are defined by $VCIBDEF in LIB.MLB and $VCIBDLLDEF in LANUDEF.MLB.

Table 4–3 describes the fields of the VCIB structure.

Table 4–3: VCIB structure

FIELD NAME	ACCESS BY UPPER VCM	ACCESS BY LAN DRIVER	DESCRIPTION
L_FLINK	W	N	Forward queue link
L_BLINK	W	N	Back queue link
W_SIZE	RI	N	Size of structure
B_TYPE	RI	R	Type of structure DYN$C_DECNET
B_SUB_TYPE	RI	R	Subtype of structure DYN$C_NET_VCI_VCIB
L_VCI_ID	RI	R	VCI ID of the upper VCM
W_VERSION_UPPER	RI	R	VCI version of the upper VCM
W_VERSION_LOWER	R	RI	VCI version of the LAN driver
A_PORTMGMT_INITIATE	R	I	Address of Port Management Initiate routine
A_PORTMGMT_COMPLETE	I	R	Address of Port Management Complete routine
A_TRANSMIT_INITIATE	R	I	Address of Transmit Initiate routine
A_TRANSMIT_COMPLETE	I	R	Address of Transmit Complete routine
A_RECEIVE_COMPLETE	I	R	Address of Receive Complete routine
A_REPORT_EVENT	I	R	Address of Report Event routine
A_DLL_INPUT_LIST	I	R	Address of input item list
(W)L_DLL_CLIENT_FLAGS	I	R	Flags specified by upper VCM
(W)L_DLL_FLAGS	R	I	Flags specified by LAN driver
(W)L_DLL_TYPE	R	I	Type of medium
(W)L_DLL_HDR_SIZE	R	I	Maximum header size in bytes
(W)L_DLL_XMT_SIZE	R	I	Maximum transmit size in bytes
(W)L_DLL_CHAIN_SIZE	R	I	Minimum size of 1st entry in chain in bytes
A_LAN_BUILD_HDR	R	I	Address of Build Header routine
A_LAN_TRANSMIT_FRAME	R	I	Address of Transmit Frame routine
A_LAN_TRANSMIT_AVAIL	R	I	Address of longword containing the number of transmit entries available on the device
(W)L_LAN_MAX_RCV	I	R	Maximum number of outstanding receives allowed

Field name

If the field name begins with a letter in parentheses, then the letter in the parentheses is used in the OpenVMS VAX name.

Key to Access

C: Constant value.
N: This module has no access to this field.
W: The module has read and write access to the field.
R: The module has read access to the field.
I: This field is initialized by this module, this does not imply that the module has access to the field after initialization.

Table 4–3 (Cont.): VCIB structure

FIELD NAME	ACCESS BY UPPER VCM	ACCESS BY LAN DRIVER	DESCRIPTION
(W)L_LAN_OUT_RCV	N	IW	Number of outstanding receives
(W)L_LAN_CLIENT_FLAGS	I	R	Flags specified by upper VCM
(W)L_LAN_FLAGS	R	I	Flags specified by LAN driver
A_LAN_LDC	R	I	Address of LDC structure
K_LAN_FIXED_LENGTH	C	C	Length of the required portion of the VCIB

Field name

If the field name begins with a letter in parentheses, then the letter in the parentheses is used in the OpenVMS VAX name.

Key to Access

C: Constant value.
N: This module has no access to this field.
W: The module has read and write access to the field.
R: The module has read access to the field.
I: This field is initialized by this module, this does not imply that the module has access to the field after initialization.

A detailed description of each of these VCIB fields follows:

- VCIB$L_FLINK

  This field is used only by the upper VCM. It may be used to put the VCIB into a queue or for private storage.

- VCIB$L_BLINK

  This field is used only by the upper VCM. It may be used to put the VCIB into a queue or for private storage.

- VCIB$W_SIZE

  This field is used only by the upper VCM. It should be used to hold the size of the VCIB so it can be used during deallocation of the VCIB.

- VCIB$B_TYPE

  The upper VCM sets this field to the value DYN$C_DECNET (defined in $DYNDEF in LIB.MLB).

- VCIB$B_SUB_TYPE

  The upper VCM sets this field to the value DYN$C_NET_VCI_VCIB (defined in $DYNDEF in LIB.MLB).

- VCIB$L_VCI_ID

  The upper VCM sets this field to its VCI ID. The VCI IDs are defined in $VCIBDEF in LIB.MLB. Note that there are some VCI IDs for customer use. Table A–1 lists the registered VCI IDs.

- VCIB$W_VERSION_UPPER

For this version of the VCI, the upper VCM sets this field to the value 1.

- VCIB$W_VERSION_LOWER

  For this version of the VCI, the LAN driver sets this field to the value 1.

- VCIB$A_PORTMGMT_INITIATE

  The LAN driver stores the address of the port management initiate routine in this field.

- VCIB$A_PORTMGMT_COMPLETE

  The upper VCM stores the address of the port management complete routine in this field.

- VCIB$A_TRANSMIT_INITIATE

  The LAN driver stores the address of the transmit initiate routine in this field.

- VCIB$A_TRANSMIT_COMPLETE

  The upper VCM stores the address of the transmit complete routine in this field.

- VCIB$A_RECEIVE_COMPLETE

  The upper VCM stores the address of the receive complete routine in this field.

- VCIB$A_REPORT_EVENT

  The upper VCM stores the address of the report event routine in this field.

- VCIB$A_DLL_INPUT_LIST

  This field is initialized by the upper VCM to contain the address of an input item list header (or the value zero). The item list is used by the LAN driver to determine which LAN device the upper VCM is trying to create a VCI port to. If this field is zero, if the item list is empty, or if the LAN driver cannot find the device name item in the list, then the LAN driver uses the first LAN device. If the upper VCM wishes to specify a particular LAN device for the LAN driver to use, then this field must point to an item list header that contains one item; the DLL$K_LAN_DEVICE item.

  The value of the DLL$K_LAN_DEVICE item is the address of a counted string which contains the device name of the device the upper VCM would like to use. This item is used when the upper VCM knows the device name. The DLL$K_LAN_DEVICE item is passed as a longword parameter where the longword contains the address of a counted string which contains the name of the device. Currently, all CSMACD and FDDI devices require a three character device name. Note that the LAN$GET_DEVICE routine returns the address of a counted string name of a device. This address can be used as the DLL$K_ LAN_DEVICE item list value.

  DLL$K_LAN_DEVICE is defined in $LANUDEF in LANUDEF.MLB.

  Note that the item list structures are define in Section 4.6.

- VCIB$W_DLL_CLIENT_FLAGS

  This field contains flags that are initialized by the upper VCM. In this version, there are no flags defined in this field. The upper VCM sets this field to zero.

- VCIB$W_DLL_FLAGS

  This field contains flags that are initialized by the LAN driver. The flags for this field include the following.

  - VCIB$V_DLL_XMT_CHAIN - Initialized before PORT_USABLE. Set if the LAN driver supports transmit chaining on this device, else cleared. If set, the upper VCM is not allowed to issue a transmit chain request to the LAN driver. Since OpenVMS AXP will support transmit chaining on all devices, this flag will never be set on OpenVMS AXP.

On OpenVMS VAX, if this flag is set, the upper VCM is not allowed to issue transmit chain requests to the LAN driver.

- VCIB$W_DLL_TYPE

  This field is used by the upper VCM to know which type of medium it is using. Possible values for this field include:

  - VCIB$K_DLL_CSMACD
  - VCIB$K_DLL_FDDI

- VCIB$W_DLL_HDR_SIZE

  This field is used by the upper VCM on transmit requests. Before the PORT_USABLE event, the LAN driver initializes this field to be the maximum number of bytes the LAN driver may need for its transmit header. The upper VCM must leave at least this much space before the data in its transmit VCRPs for the LAN header. Note that the LAN header cannot be backfilled into the scratch area of the VCRP. The LAN driver is not required to ensure that it does not backfill its header into the scratch area of the VCRP. The upper VCM must ensure that it leaves enough space for the LAN driver to backfill its header without going into the scratch area of the VCRP.

  Note that the constant LAN$C_MAX_HDR_SIZE can also be used by the upper VCM. This constant represents the maximum header size for all LAN drivers.

- VCIB$W_DLL_XMT_SIZE

  This field is used by the upper VCM on transmit requests. The LAN driver initializes this field before it reports the PORT_USABLE event. The LAN driver initializes this field to be the maximum number of bytes the upper VCM is allowed to transmit per transmit request. The LAN driver is not required to ensure that the upper VCM does not transmit more than the allowable number of bytes. Note that this size does not include the LAN header. The contents of this field are invalid after the upper VCM has disabled the port and after the LAN driver has disabled the port. The field is again valid after the port is re-enabled and the upper VCM receives the PORT_USABLE event.

- VCIB$W_DLL_CHAIN_SIZE

  This field is used by the upper VCM on chained transmit requests. Before the PORT_ USABLE event, the LAN driver initializes this field to be the minimum number of bytes required in the first buffer of a chained transmit. That is, this field is set to the minimum number of bytes that must exist in the VCRP portion of a chained request. This number includes the LAN header. The upper VCM is required to ensure that at least this number of bytes are available in the VCRP of a chained request.

  The upper VCM may provide further optimization by ensuring that the first buffer contains at least this much data; instead of forcing the LAN driver to copy data from the second buffer into this buffer. The performance improvement is realized when the upper VCM copies the appropriate number of bytes from the second buffer into the VCRP. It may cost the LAN driver more CPU time to do the copy if the LAN driver has to initialize system PTEs to perform the copy. The upper VCM may be able to do the copy when the process is still current and the buffer is still available through process space.

- VCIB$A_LAN_BUILD_HDR

  The LAN driver stores the address of the build header routine in this field.

- VCIB$A_LAN_TRANSMIT_FRAME

  The LAN driver stores the address of the transmit frame routine in this field.

- VCIB$A_LAN_TRANSMIT_AVAIL

  This field is a pointer to a longword containing the number of transmit entries available on the device. If an upper VCM is using multiple devices, this can be used to determine which device is less/more busy.

- VCIB$W_LAN_MAX_RCV

  The upper VCM initializes this word to contain the maximum number of outstanding receives the upper VCM wishes to have. This field is only used if the VCIB$V_LAN_RCV_LIM flag is set to 1.

- VCIB$W_LAN_OUT_RCV

  This field is used solely by the LAN driver to keep track of the number of outstanding receives. This field is only used if the VCIB$V_LAN_RCV_LIM flag is set to 1.

- VCIB$W_LAN_CLIENT_FLAGS

  This field contains flags that are initialized by the upper VCM. All remaining bits of this flag field must be set to zero.

  - VCIB$V_LAN_RCV_LIM - If set to 1, the upper VCM wants the LAN driver to limit the number of outstanding receive VCRPs being held by the upper VCM. The VCIB$W_LAN_MAX_RCV and VCIB$W_LAN_OUT_RCV fields are used to implement this feature. If set to 0, the upper VCM can have an infinite number of outstanding receives.
  - VCIB$V_LAN_RCV_ICS - Initialized to 0.
  - VCIB$V_LAN_SFR - Skip field restore - This flag is ignored if the VCIB$V_LAN_FTC flag is set to 1. If FTC=0 and SFR=0, the LAN driver will save the BOFF, BCNT, and TOTAL_PDU_SIZE fields in the VCRP of transmit requests. That is, these fields are not destroyed by the LAN driver. This flag is initialized before the upper VCM creates the VCI port. If FTC=1 or SFR=1, then these fields may be destroyed by the LAN driver before the VCRP is returned to the upper VCM.
  - VCIB$V_LAN_SCC - Skip completion check - The upper VCM sets this flag to 1 if the VCIB$V_LAN_FTC flag is set to 0. This flag is ignored if the VCIB$V_LAN_FTC flag is set to 1.
  - VCIB$V_LAN_FTC - Fast transmit complete - If set to 1, the upper VCM wishes to have fast transmit completion. If set to 1, the LAN driver will not restore the BOFF, BCNT, and TOTAL_PDU_SIZE fields of transmit requests.

- VCIB$W_LAN_FLAGS

  This field contains flags that are initialized by the LAN driver. In this version, there are no flags defined for this field.

- VCIB$A_LAN_LDC

  The address of the LDC structure. The upper VCM can examine the fields in the LDC structure; but cannot write the LDC fields. The LDC structure is defined in Section 4.2.

- VCIB$K_LAN_FIXED_LENGTH

  This is a constant representing the minimum size of the VCIB required for the LAN drivers.

4.4 VCRP

The VCRP is the data structure used to pass requests between two VCMs.

The VCRP contains request information and the completion status of that request. There are basically two types of VCRPs; management VCRPs and data VCRPs. Port Management requests use management VCRPs. Management VCRPs contain a section that allows parameter buffers to be passed to the LAN driver. Transmit and Receive requests use data VCRPs. Data VCRPs contain a section that allow them to pass a packet between the VCMs. For transmit VCRPs, the packet may be contained within the VCRP or attached to the VCRP as a buffered or a direct I/O request.

The registered name for this data structure is VCRP$.

The VCRP fields are defined by $VCRPDEF in LIB.MLB and $VCRPLANDEF in LANUDEF.MLB.

Table 4–4 describes the fields of the VCRP structure.

Table 4–4: VCRP structure

	ACCESS ON			
FIELD NAME	**MGMT**	**XMIT**	**RECV**	**DESCRIPTION**
L_FLINK	D	D	D	Forward queue link
L_BLINK	D	D	D	Back queue link
W_SIZE	UO	UO	LO	Size of structure
B_TYPE	UI	UI	LI	Type of structure DYN$C_VCRP
B_RMOD	UO	UO	UO	ACB - Request modifier
L_PID	UO	UO	UO	ACB - Process Identifier
A_ASTADR	UO	UO	UO	ACB - Address of user AST routine
L_ASTPRM	UO	UO	UO	ACB - Parameter for user AST routine
A_KAST	UO	UO	UO	ACB - Address of special kernel AST routine
W_COMMON_FLAGS	UI	UI	LI	Common flags for the request

Field name

If the field name begins with a letter in parentheses, then the letter in the parentheses is used in the OpenVMS VAX name.

Key to Access

C: Constant value.
A: Field is accessible to the current owner of the VCRP, but its semantics are well defined and must be followed.
D: Field is owned and can be destroyed by the current owner of the VCRP.
E: This field doesn't exist for this type of request.
LI: The LAN driver initializes this field and the upper VCM can read it.
UI: The upper VCM initializes this field and the LAN driver can read it.
UD: The upper VCM initializes this field and the LAN driver can set it to a default value.
LO: Only the LAN driver has access to this field.
UO: Only the upper VCM has access to this field.

Table 4–4 (Cont.): VCRP structure

FIELD NAME	ACCESS ON MGMT	XMIT	RECV	DESCRIPTION
B_FLAGS	UO	UO	UO	Request-specific flags
B_MODE	UO	UO	UO	Access mode - currently unused
A_DEALLOC_RTN	UO	UO	LI	Deallocation routine address
L_FUNCTION	UI	UI	LI	Function code
L_ASSOCIATION_ID	UO	UO	UO	Context for the creator of the VCRP
L_CONNECTION_ID	UO	UO	UO	Context for the creator of the VCRP
Q_REQUEST_STATUS	LI	LI	LI	Completion status of request
A_CREATOR	UO	UO	UO	Usually the VCIB address of the VCRP
Q_CREATOR_DATA	UO	UO	LO	Context for the creator of the VCRP
A_STACK	UO	UO	UO	Pointer to the context stack area
T_INTERNAL_STACK	UO	UO	UO	Internal context area
A_LAN_P2BUFF	UI	E	E	Address of address of P2 buffer
L_LAN_P2BUFF_SIZE	UI	E	E	Number of bytes in P2 buffer
A_DCB_LINK	E	UI	UO	Address of first DCBE
L_BOFF	E	UI	LI	Offset to data
L_BCNT	E	UI	LI	Number of bytes of packet in the VCRP
L_TOTAL_PDU_SIZE	E	UI	E	Number of bytes in all segments
(B)L_LAN_FC	E	UI	LI	FC field value
Q_LAN_T_DEST	E	UI	E	Destination address
L_LAN_802XMT	E	UD	E	802 only - overlay of next 3 fields
W_LAN_T_CTL	E	UD	E	802 only - control field value
B_LAN_T_CTL_SIZE	E	UD	E	802 only - control field value size
B_LAN_T_DSAP	E	UD	E	802 only - DSAP field value
(B)L_LAN_T_RESP	E	UD	E	802 only - response flag
A_LAN_R_HEADER	E	E	LI	Pointer to destination address of LAN header

Field name

If the field name begins with a letter in parentheses, then the letter in the parentheses is used in the OpenVMS VAX name.

Key to Access

C: Constant value.
A: Field is accessible to the current owner of the VCRP, but its semantics are well defined and must be followed.
D: Field is owned and can be destroyed by the current owner of the VCRP.
E: This field doesn't exist for this type of request.
LI: The LAN driver initializes this field and the upper VCM can read it.
UI: The upper VCM initializes this field and the LAN driver can read it.
UD: The upper VCM initializes this field and the LAN driver can set it to a default value.
LO: Only the LAN driver has access to this field.
UO: Only the upper VCM has access to this field.

Table 4–4 (Cont.): VCRP structure

FIELD NAME	ACCESS ON MGMT	XMIT	RECV	DESCRIPTION
T_SCRATCH	D	D	D	Scratch area
K_FIXED_LENGTH	C	C	C	Length of the required portion of the VCRP
T_DATA	A	A	A	Data area for data requests

Field name

If the field name begins with a letter in parentheses, then the letter in the parentheses is used in the OpenVMS VAX name.

Key to Access

C: Constant value.
A: Field is accessible to the current owner of the VCRP, but its semantics are well defined and must be followed.
D: Field is owned and can be destroyed by the current owner of the VCRP.
E: This field doesn't exist for this type of request.
LI: The LAN driver initializes this field and the upper VCM can read it.
UI: The upper VCM initializes this field and the LAN driver can read it.
UD: The upper VCM initializes this field and the LAN driver can set it to a default value.
LO: Only the LAN driver has access to this field.
UO: Only the upper VCM has access to this field.

All VCRP fields not defined in this specification should not be used by the upper VCM for any purpose.

The fields marked with access UO are not described in the following list. These fields are for the sole use of the upper VCM and the LAN driver never accesses them. Most upper VCMs will use these fields for private storage.

A detailed description of the remaining VCRP fields follows:

- VCRP$L_FLINK

  This field is used by the VCM that currently owns the VCRP. It is typically used to put the VCRP into a queue.
- VCRP$L_BLINK

  This field is used by the VCM that currently owns the VCRP. It is typically used to put the VCRP into a queue.
- VCRP$W_SIZE

  This field is initialized by the VCM that allocated the VCRP. It should be set to the size of the VCRP so it can be used during deallocation of the VCRP.
- VCRP$B_TYPE

  This field is initialized by the VCM that allocated the VCRP. It should be set to the value DYN$C_VCRP (defined in $DYNDEF in LIB.MLB).
- VCRP$W_COMMON_FLAGS

This flag field contains flags that are common to all VCRPs. The flags used in this field follow. All remaining bits of this flag field must be set to zero.

- VCRP$V_CMN_LOCKED - This flag is for transmit only. When set, this flag indicates that user data has been locked down for a direct I/O. This flag can be used by the upper VCM to keep track of direct I/O transmit requests. The LAN drivers do not examine this flag on transmit requests.
- VCRP$V_CMN_CACHE - When set indicates that this data structure is from the owner's cache.

  This flag is initialized and used only by the owner of the structure. The owner of the structure can use the flag to know if the structure was allocated from the owner's cache of data structures. This provides a simple method for the owner to keep track of the data structures being used.

  The upper VCM can use this flag in the VCRPs and DCBEs it allocates if it's useful to keep this context.
- VCRP$V_CMN_MGMT - When set indicates that this VCRP is for a management request. When clear indicates that this VCRP is for a data request. For port management requests, this flag is set to 1. For transmit and receive requests, this flag must be clear.

- VCRP$A_DEALLOC_RTN

  This field contains the address of a deallocation routine to be used by the upper VCM when it is finished with a receive VCRP. This field is only used in receive VCRPs. For receive VCRPs, the LAN driver stores the address of a completion routine in this field.

  The deallocate routine is called at IPL8 with the IOLOCK8 spinlock as the highest spinlock taken out. The address of the VCRP is in R3 and the deallocate routine can only destroy R3.

  This is the only way for an upper VCM to discard a receive VCRP - it must call the DEALLOC_RTN routine. It cannot deallocate the receive VCRP and it cannot pass it to the LAN driver any other way.
- VCRP$L_FUNCTION

  Function code for this request.
  - For port management requests, the upper VCM sets this field to either VCRP$K_FC_ENABLE_PORT, VCRP$K_FC_DISABLE_PORT, or VCRP$K_FC_LAN_CHANGE_PORT.
  - For transmit requests, the upper VCM sets this field to VCRP$K_FC_TRANSMIT.
  - For receive requests, the LAN driver sets this field to VCRP$K_FC_RECEIVE.
- VCRP$Q_REQUEST_STATUS

  This field contains the completion status for the request. See Section 3.6 for more information.
- VCRP$Q_CREATOR_DATA

  This field is used as private storage for the VCM that allocated the VCRP.
- VCRP$T_INTERNAL_STACK

  This field can be used by the upper VCM as storage. Its size is VCRP$S_INTERNAL_STACK.
- VCRP$A_LAN_P2BUFF

This field contains the address of a longword containing the address of the P2 buffer. The P2 buffer must be in system space. The P2 buffer has the same format as the QIO SETMODE!STARTUP P2 buffer. The P2 buffer is not to be modified or deallocated by the upper VCM until the LAN driver returns the VCRP to the upper VCM through the PORTMGMT_COMPLETE service routine.

For more information about which parameters can be placed in the P2 buffer, what values these parameters can have, and what the parameters mean, see the LAN chapter of the VMS I/O User's Manual.

For VCI, there are some restrictions on some of the P2 parameters. The following list describes these restrictions. In future releases, some of these restrictions may be lifted.

- NMA$C_PCLI_CRC - This parameter is passed with or defaulted to the value NMA$C_ STATE_ON. The CRC for the transmit packets cannot be supplied by the upper VCM.
- NMA$C_PCLI_PRM - This parameter is passed with or defaulted to the value NMA$C_STATE_OFF. Promiscuous mode is not supported through the VCI.
- NMA$C_PCLI_ACC - For VCI ports using Ethernet format, this parameter is passed with or defaulted to the value NMA$C_ACC_EXC. Protocol sharing is not supported through the VCI.

- VCRP$L_LAN_P2BUFF_SIZE

  This field contains the number of bytes in the P2 buffer.

- VCRP$A_DCB_LINK

  This field contains the address of a DCBE which describes the next portion of a packet in a chained request. If there are no more segments, then this field is set to zero.

- VCRP$L_BOFF

  For the VCRP, this field contains the offset (in bytes) from the beginning of the VCRP to the first byte of the data embedded in this VCRP.

  For the DCBE, this field may contain the offset from the page pointed to by the SVAPTE field to the beginning of the data.

- VCRP$L_BCNT

  This field contains the number of bytes in the data segment embedded in the VCRP.

- VCRP$L_TOTAL_PDU_SIZE

  This field contains the number of bytes in the entire packet including the embedded data and the data in all the DCBEs chained to this VCRP.

- VCRP$B_LAN_FC

  For receive requests, this field contains the FC field value from the FDDI packet and contains the value zero for CSMACD packets.

  For FDDI transmit requests, this field is used if the XFC parameter of the P2 buffer was set to the value zero. In this case, this field should contain the priority bits (the low order three bits) to be used in the FC field of the packet to be transmitted.

- VCRP$Q_LAN_T_DEST

  The first six bytes of this field contain the destination address. The last two bytes are unused by the LAN driver.

- VCRP$L_LAN_802XMT

This field is an overlay of the fields VCRP$W_LAN_T_CTL, VCRP$B_LAN_T_CTL_SIZE, and VCRP$B_LAN_T_DSAP. The use of these fields and this overlay is described in the following paragraph.

The following four fields are initialized by the upper VCM on transmit requests only if the port is an 802 port. If the CTL, CTL_SIZE, and DSAP fields are all zero, then all four fields are defaulted. If they are defaulted, then the default values are placed in those fields by the LAN driver. This implies that these fields are only destroyed by the LAN driver if they are defaulted. If they are destroyed, they will contain the default values when the VCRP is returned to the upper VCM through the TRANSMIT_COMPLETE service routine. Note that the VCRP$W_LAN_T_CTL, VCRP$B_LAN_T_CTL_SIZE, and VCRP$B_LAN_T_DSAP fields are contiguous (in that order) in the VCRP and that they can be referenced as a longword using the field named VCRP$L_LAN_802XMT.

- VCRP$W_LAN_T_CTL

  If specified, this field contains the 802 control field value to be used by the LAN driver. The value is either one byte or two bytes. The next field (VCRP$B_LAN_T_CTL_SIZE) contains the number of bytes in this field. If this field is one byte, it's passed in the low order byte. If this field is defaulted, it is defaulted to the value 3 (Unnumbered Information).

- VCRP$B_LAN_T_CTL_SIZE

  If specified, this field is used by the LAN driver to detect the size of the control field value in the VCRP$W_LAN_T_CTL field. If the low bit is clear, the CTL field is 2 bytes. If the low bit is set, the CTL field is 1 byte. If this field is defaulted, it's defaulted to the value 1.

- VCRP$B_LAN_T_DSAP

  If specified, this field contains the value to be used in the DSAP field of the frame to be transmitted. If this field is defaulted, it's defaulted to the SAP of the upper VCM's 802 port.

- VCRP$B_LAN_T_RESP

  This field tells the LAN driver whether it should transmit a response frame or a command frame. If the low order bit is 1, then a response frame is transmitted. If the low order bit is 0, then a command frame is transmitted.

- VCRP$A_LAN_R_HEADER

  This field contains the address of the destination address field of the LAN header of the frame received. The upper VCM can examine the header for the destination address and the source address. The destination address is always the first 6 bytes of the header and the source address is always the next 6 bytes of the header.

- VCRP$T_SCRATCH

  This field is scratch area and may be completely destroyed by the current owner of the VCRP. The size of the scratch area is VCRP$K_SCRATCH_AREA_LENGTH.

- VCRP$K_FIXED_LENGTH

  This is a constant which represents the minimum size VCRP excluding the space required for the data embedded in transmit and receive VCRPs.

- VCRP$T_DATA This field represents the beginning of where packets may be placed. Remember that for transmit requests, the upper VCM has to leave enough space after this offset for the LAN header of the packet. The value stored in the field VCIB$W_DLL_ HDR_SIZE represents the minimum size of the LAN header for this VCI port.

4.5 DCBE

The DCBE is used to connect a buffer containing part of a packet to a VCRP. In the past, the data portion of a packet had to be copied into the system data structure being used by the LAN driver. To prevent this copying from occurring (to save CPU cycles), the DCBE was created. This means that with the VCRP and the DCBE, one packet can span multiple buffers. This is called chaining. The term chaining is used to represent connecting a DCBE to a VCRP where the DCBE points to part of a packet. Note that the LAN header is always contained in the VCRP, and never in a DCBE.

Chaining is only possible in transmit VCRPs. Receive VCRPs always contain all of the packet within the VCRP and Management VCRPs do not have any fields that are allowed to point to a DCBE. Transmit VCRPs use the field VCRP$A_DCB_LINK to point to a DCBE.

The registered name for this data structure is DCBE$.

The DCBE fields are defined by $DCBEDEF in LIB.MLB.

Table 4–5 describes the fields of the DCBE structure.

Table 4–5: DCBE structure

FIELD NAME	ACCESS	DESCRIPTION
L_FLINK	D	Forward queue link
L_BLINK	D	Back queue link
W_SIZE	UO	Size of structure
B_TYPE	UI	Type of structure DYN$C_DECNET
B_SUB_TYPE	UI	Subtype of structure DYN$C_VCI_DCB
W_COMMON_FLAGS	UI	Common flags for the request
B_FLAGS	UO	Request-specific flags
B_MODE	UO	Access mode - currently unused
A_DEALLOC_RTN	UO	Deallocation routine address
A_DCB_LINK	UI	Address of next DCBE
L_SVAPTE	UI	System virtual address of PTE of data

Key to Access

C: Constant value.
D: Field is owned and can be destroyed by the current owner of the DCBE.
UI: The upper VCM initializes this field and the LAN driver can read it.
UO: Only the upper VCM has access to this field.

Table 4–5 (Cont.): DCBE structure

FIELD NAME	ACCESS	DESCRIPTION
L_BUFFER_ADDRESS	UI	System virtual address of data
L_BOFF	UI	Offset to data
L_BCNT	UI	Number of bytes of data in this buffer
K_DCB_HEADER	C	Length of the required portion of the DCBE

Key to Access

C: Constant value.
D: Field is owned and can be destroyed by the current owner of the DCBE.
UI: The upper VCM initializes this field and the LAN driver can read it.
UO: Only the upper VCM has access to this field.

The fields marked with access UO are not described in the following list. These fields are for the sole use of the upper VCM and the LAN driver never accesses them. A detailed description of the remaining DCBE fields follows:

- DCBE$L_FLINK

  This field is used by the VCM that currently owns the DCBE. It is typically used to put the DCBE into a queue.
- DCBE$L_BLINK

  This field is used by the VCM that currently owns the DCBE. It is typically used to put the DCBE into a queue.
- DCBE$W_SIZE

  This field is initialized by the VCM that allocated the DCBE. It should be set to the size of the DCBE so it can be used during deallocation of the DCBE.
- DCBE$B_TYPE

  This field is initialized by the VCM that allocated the DCBE. It should be set to the value DYN$C_DECNET (defined in $DYNDEF in LIB.MLB).
- DCBE$B_SUB_TYPE

  This field is initialized by the VCM that allocated the DCBE. It should be set to the value DYN$C_VCI_DCB (defined in $DYNDEF in LIB.MLB).
- DCBE$W_COMMON_FLAGS

  See the description of VCRP$W_COMMON_FLAGS in Section 4.4.
- DCBE$A_DCB_LINK

  The field DCBE$A_DCB_LINK is always set to zero.
- DCBE$L_SVAPTE

  This field may contain the system virtual address of the page table entry that points to the first page of the segment.
- DCBE$L_BUFFER_ADDRESS

  This field may contain the system virtual address of the first byte of the data.
- DCBE$L_BOFF

See the description of VCRP$L_BOFF in Section 4.4.

- DCBE$L_BCNT

  This field contains the number of bytes in the data segment pointed to by the DCBE.

- DCBE$S_DCBEDEF

  This is a constant representing the minimum size of the DCBE.

4.6 LIL

The LAN Item List (LIL) is the data structure used to pass the device name to the LAN driver in the call to VCI$LAN_CREATE_PORT. A LAN Item List contains a header pointing to a series of items; so there are really two structures. One structure defines the item list header, and the other structure defines an item in the item list. Note that the items in the item list may actually follow the header in virtually contiguous memory or may be placed in a separate memory buffer. The items in the list, however, are virtually contiguous with each other. The fields in these structures begin with LIL$.

The LIL fields are defined by $LILDEF in LANUDEF.MLB.

Table 4–6 describes the fields of the LIL header structure.

Table 4–7 describes the fields of the LIL item structure.

Table 4–6: LIL header structure

FIELD NAME	DESCRIPTION
L_LISTLEN	Number of bytes in the list of items
A_LISTADR	Address of the first byte in the list of items
W_SIZE	Size of memory to be deallocated
B_TYPE	Type of structure DYN$C_DECNET
B_SUBTYPE	Subtype of structure DYN$C_NET_ITEM
T_DATA	Place where the items can begin

- LIL$L_LISTLEN

  This field contains the size (in bytes) of the items; not including the header (not including the fields up to and including LIL$B_SUBTYPE).

- LIL$A_LISTADR

  This field contains the address of the list of items. This is usually the address of the LIL$T_DATA field.

- LIL$W_SIZE

This field contains the size (in bytes) of this structure (from the LISTLEN field to the end of the DATA section). If the item (DATA) section is part of the allocated memory, then this size field should include that section also. This field is typically used when deallocation occurs.

- LIL$B_TYPE

  This field contains the type of structure. This field should contain the value DYN$C_ DECNET (defined in $DYNDEF in LIB.MLB).

- LIL$B_SUBTYPE

  This field contains the type of structure. This field should contain the value DYN$C_NET_ ITEM (defined in $DYNDEF in LIB.MLB).

- LIL$T_DATA

  This field marks the position in this structure where the items in the list can be placed.

  In general, the items in the list can (and usually do) begin at the LIL$T_DATA field. However, the item list header allows the items to be placed in a separate buffer.

Table 4–7: LIL item structure

FIELD NAME	DESCRIPTION
W_ITEMLEN	Number of bytes in this item
W_ITEMTAG	Tag (identifier) for this item
T_ITEMVAL	Value for this item

- LIL$W_ITEMLEN

  This field contains the number of bytes in the item. This includes the ITEMLEN, ITEMTAG, and ITEMVAL fields; so the smallest number this field can contain is 4 (in the case where the size of the value field is zero bytes).

- LIL$W_ITEMTAG

  This field contains the tag for the item. The tag is used to identify the item.

- LIL$T_ITEMVAL

  The value for the item is stored in this field. The number of bytes required to store the value and the format of the value depends on the item.

Chapter 5

FEATURES

This chapter describes some of the features available to the users of the LAN VCI. Many of these features are available through the use of particular fields in the data structures.

5.1 PERFORMANCE

This section highlights the features that provide the highest performance (lowest number of CPU cycles per I/O request).

- Use the Fast Transmit Complete feature by setting the VCIB$V_LAN_FTC flag in the VCIB$W_LAN_CLIENT_FLAGS field of the VCIB. This allows the LAN driver to use the most efficient method to complete transmit requests to the upper VCM.
- The upper VCM should not check if a receive VCRP has a success status in the VCRP$Q_ REQUEST_STATUS field. The LAN driver only completes receive VCRPs with success status values to the upper VCM.
- Use the build header routine to get a LAN header. In most cases, this routine can be called once during the life of the VCI port. Once the upper VCM gets the LAN header, the upper VCM should pre-pended it to all packets it transmits and use the transmit frame service routine. This saves CPU cycles because the LAN driver doesn't have to build a LAN header for every packet transmitted; it only needs to initialize the destination address field, possibly a length field, and (for FDDI only) the FC field (if the P2 parameter XFC was set to zero on the enable port request).
- The upper VCM can prevent copying transmit data by using the transmit chaining feature. By not copying the data, fewer CPU cycles are used.

5.2 CONVERTING A VCRP INTO A DCBE

The DCBE is smaller than the VCRP. And the fields that exist in the DCBE are exactly the same as the fields of the VCRP. That is, the DCBE fields have the same name and position as the corresponding field in the VCRP. This allows multiple VCRPs that are received at different times to be easily connected to form a single VCRP and many DCBEs. Even though the DCBEs in this case are larger than normal, they may still be called DCBEs.

5.3 CONVERTING A VCRP INTO AN ACB

The VCRP contains some fields that can be used to make the beginning of the VCRP look like an ACB (AST Control Block). These fields are generally used by upper VCMs that pass the VCRP to the I/O executive to be completed to a process. If the VCRP is not going to be passed to the I/O executive, then the ACB fields can be used by the upper VCM as permanent or temporary storage which are not destroyed by the LAN driver. The ACB fields in the VCRP include the following fields.

VCRP$B_RMOD
VCRP$L_PID
VCRP$A_ASTADR
VCRP$L_ASTPRM
VCRP$L_KAST

Chapter 6

PROCESSING

This chapter describes the steps that the upper VCM should use in making VCI requests to the LAN driver.

6.1 STARTING A VCI PORT

There are multiple steps required to getting a VCI port ready for data transfer. In general, a VCI port is created, the VCI port is enabled, and the upper VCM waits for the PORT_USABLE event.

Once the upper VCM has received a PORT_USABLE event for the port, the port can be used for data transfer.

To start a VCI port, the upper VCM performs the following steps.

1. The upper VCM allocates and initializes the VCIB that will represent this VCI port. Remember that the VCIB allocated can be larger than required by the LAN driver. The upper VCM can extend the VCIB for its own purpose.

   If the upper VCM does not know which LAN device to use, it can call the LAN$GET_ DEVICE routine to determine which LAN devices are available.

2. The upper VCM now creates the VCI port by calling the LAN driver's CREATE_PORT routine. This is a synchronous call. So the upper VCM gets status on this request when it gets control back from the call to the CREATE_PORT routine. If an error is returned, then the upper VCM should either not attempt to use that device or it could wait and try the device again once per minute (or so). The decision should be based on the error returned from the LAN driver. Some errors indicate that the device will never be usable.

   If successful, the VCIB must be kept intact by the upper VCM. It cannot be deallocated until the VCI port has been deleted.

3. As soon as control is returned to the upper VCM (from its call to the LAN driver's create port routine) and the upper VCM has checked that the status of the call was successful, the upper VCM should allocate and initialize a VCRP for the enable port request.

   The ENABLE_PORT VCRP requires a P2 buffer. The upper VCM needs to allocate and initialize a P2 buffer. The field VCRP$A_LAN_P2BUFF points to the P2 buffer.

4. With the enable port VCRP ready, the upper VCM calls the LAN driver's port management routine to enable the VCI port. The purpose of this step is to have the upper VCM request that the VCI port be enabled for use and to have the upper VCM specify the appropriate characteristics for the VCI port.
5. Remember that the enable port request is completed asynchronously. So when control is returned to the upper VCM, it should dismiss its fork thread.
6. When the upper VCM is called at its port management complete routine with its enable port request, it should examine the status of the request in the VCRP$Q_REQUEST_ STATUS field of the VCRP. If there was an error, the upper VCM should handle it as appropriate for the upper VCM.

   This is where the upper VCM would typically deallocate the VCRP and the P2 buffer used for the ENABLE_PORT request.
7. If the status of the enable port request was successful, then the upper VCM waits for a call to its report event routine with the port usable event. Once receiving this event, the upper VCM can now transmit packets and can now receive packets.

6.2 CHANGING A VCI PORT

There are only a few P2 parameters that may be changed on a VCI port that is usable. The parameters that may be changed are listed in the LAN chapter of the VMS I/O User's Manual.

To change any of these parameters on a VCI port that's usable, the upper VCM performs the following steps.

1. The upper VCM should allocate and initialize a VCRP for the change port request. The VCRP should be initialized the same as an enable port request except for putting the new parameter values in the P2 buffer and setting the function code to VCRP$K_FC_LAN_ CHANGE_PORT.
2. With the change port VCRP ready, the upper VCM calls the LAN driver's port management routine to change the VCI port.
3. The change port request is completed asynchronously. So when control is returned to the upper VCM, it should dismiss its fork thread.
4. When the upper VCM is called at its port management complete routine with its change port request, it should examine the status of the request in the VCRP$Q_REQUEST_ STATUS field of the VCRP. If there was an error, the upper VCM should handle it as appropriate for the upper VCM.
5. There is no event associated with the change port request. Once the change port VCRP has been returned to the upper VCM, the request has been completed.

6.3 SHUTTING DOWN A VCI PORT

There are multiple steps required to shutting down a VCI port. In general, the VCI port is disabled, the upper VCM waits for the PORT_UNUSABLE event, and the VCI port is deleted.

To shut down a VCI port, the upper VCM performs the following steps.

1. The upper VCM should allocate and initialize a VCRP for the disable port request. The VCRP should be initialized the same as an enable port request except that the P2 buffer fields are set to zero and the function code is set to VCRP$K_FC_DISABLE_PORT.
2. With the disable port VCRP ready, the upper VCM calls the LAN driver's port management routine to disable the VCI port.
3. Remember that the disable port request is completed asynchronously. So when control is returned to the upper VCM, it should dismiss its fork thread.
4. Before the disable port request is completed, the LAN driver completes all outstanding VCI requests to the upper VCM with the status SS$_ABORT or another status. The status returned depends on the state of the outstanding request.
5. When the upper VCM is called at its port management complete routine with its disable port request, it should examine the status of the request in the VCRP$Q_REQUEST_ STATUS field of the VCRP. If there was an error, the upper VCM should handle it as appropriate for the upper VCM.
6. If the status of the disable port request was successful, then the upper VCM waits for a call to its report event routine with the port unusable event.
7. Before deleting the VCI port, the upper VCM ensures that all outstanding receive VCRPs have been returned to the LAN driver. Failure to do so may cause the system to crash.
8. The upper VCM may now delete the VCI port by calling the LAN driver's DELETE_PORT routine. This is a synchronous call. So the upper VCM gets status on this request when it gets control back from the call to the DELETE_PORT routine.
9. The upper VCM may now deallocate the VCIB.

6.4 TRANSMITTING A PACKET

There are multiple options available to the upper VCM to transmit packets. The packets can be transmitted from the upper VCM with or without the LAN header as part of the request. The packets can be transmitted chained or unchained. In a chained request, the second segment in the chain can be pointed by a system virtual address or by a SVAPTE/BOFF set. This section describes how to use and specify these options.

6.4.1 TRANSMIT FRAME

For the upper VCM to include the LAN header as part of the transmit request, the upper VCM first gets the LAN header. Note that the LAN header returned to the upper VCM does NOT have all the fields initialized. In particular, the destination address field is not initialized in the header. The remaining fields are initialized on the actual transmit request. This allows most upper VCMs to request one header from the LAN driver, to save that header, and to use that header on every transmit request - even though the destination address on each transmit

request may be different. The build header routine is called to have the LAN driver return a LAN header to the upper VCM.

If the upper VCM does not wish to include the LAN header as part of its transmit request, then it calls the transmit initiate routine. Some of the VCRP fields to be initialized by the upper VCM when making a transmit request with no LAN header include:

VCRP$L_BOFF is set to the offset from the beginning of the VCRP to the upper VCM's data in the VCRP. Remember to leave enough space for the LAN header.
VCRP$L_BCNT is set to the number of bytes of data in the VCRP.
VCRP$L_TOTAL_PDU_SIZE is set to VCRP$L_BCNT (plus DCBE$L_BCNT if this is a chained request).
VCRP$Q_LAN_DEST is set to the destination address.
VCRP$L_LAN_802XMT and VCRP$B_LAN_T_RESP are initialized if the VCI port has been started with the 802 format.
VCRP$B_LAN_FC is set to the priority bits of the FC field if this an FDDI device and the upper VCM set the XFC P2 parameter to zero.

If the upper VCM includes the LAN header as part of its transmit request, then it calls the transmit frame routine. The LAN header is always placed in the VCRP. Some of the VCRP fields to be initialized by the upper VCM when making a transmit request with a LAN header include:

VCRP$L_BOFF is set to the offset from the beginning of the VCRP to the LAN header that has been placed in the VCRP.
VCRP$L_BCNT is set to the number of bytes of data in the VCRP (for this case, it includes the LAN header).
VCRP$L_TOTAL_PDU_SIZE is set to VCRP$L_BCNT (plus DCBE$L_BCNT if this is a chained request)
VCRP$Q_LAN_DEST is set to the destination address.
VCRP$B_LAN_FC is set to the priority bits of the FC field if this an FDDI device and the upper VCM set the XFC parameter to zero.

When issuing a transmit request with the LAN header already included, the LAN driver requires that the LAN header be placed in the VCRP following the VCRP$T_DATA field. The number of bytes required for the LAN header varies across the different LAN mediums and the different packet formats. The LAN driver stores the minimum space required for each VCI port's LAN header in the field VCIB$W_DLL_HDR_SIZE. The upper VCM can also use the constant value LAN$C_MAX_HDR_SIZE. Using this constant makes it possible for the upper VCM to use all transmit VCRPs across all LAN devices. The best way for the upper VCM to accommodate this requirement is for the upper VCM to use the value VCRP$T_DATA plus LAN$C_MAX_HDR_SIZE when allocating its transmit VCRPs. If more space is required for the LAN header plus any data from the upper VCM, then a larger allocation size can be used.

Note that the size required by the LAN driver (VCIB$W_DLL_HDR_SIZE) may be larger than the size of the header returned from the build header routine. The value returned by the build header routine should NOT be used as the amount of space needed by the LAN driver after the VCRP$T_DATA field. The size returned from the build header routine should ONLY be used to copy the LAN header into the transmit VCRP.

6.4.2 TRANSMIT CHAINING

The transmit chaining feature of the VCI can provide large CPU cycle gains. This feature allows the upper VCM to leave the user data in its own memory instead of copying the user data into the VCRP.

The basic rule for transmit is that the upper VCM must allow enough space after the VCRP$T_DATA field for the LAN header. The number of bytes required by the LAN driver is supplied to the upper VCM in the field VCIB$W_DLL_HDR_SIZE. Or the upper VCM can use the constant LAN$C_MAX_HDR_SIZE.

The two supported transmit request formats are:

- UNCHAINED - In this format there is a VCRP with no attached buffer or DCBE. All the data to be transmitted is in the VCRP.
- CHAINED - In this format there is a VCRP with embedded data and a DCBE with attached data. In this case, the user data embedded in the VCRP can consist of zero bytes; but the upper VCM still initializes the VCRP$L_BCNT and VCRP$L_BOFF fields of the VCRP.

To set up an unchained request, the upper VCM uses one VCRP and no DCBEs.

The following list describes the steps required to set up the fields for an unchained request. Other fields are also required; but these are the fields that are different for chained and unchained transmit requests.

- The packet is placed into the VCRP. There are usually three parts to the packet; the LAN header, the upper VCM's header, and the data. The upper VCM's header begins at VCRP$T_DATA + header size (where header size is either LAN$C_MAX_HDR_SIZE or the contents of VCIB$W_DLL_HDR_SIZE). The upper VCM copies its header and the data into the VCRP. If it has the LAN header (from a previous call to BUILD_HDR), then the upper VCM can copy the LAN header into the VCRP; placing it just before upper VCM's header.
- Set VCRP$L_BOFF to be the number of bytes to either the LAN header (if the LAN header has been copied into the VCRP) or to the upper VCM's header.
- Set VCRP$L_BCNT to be equal to the number of bytes that have been placed in the VCRP; including the LAN header if it has been copied into the VCRP.
- Set VCRP$L_TOTAL_PDU_SIZE to the value in VCRP$L_BCNT.
- Set VCRP$A_DCB_LINK to zero.

To chain a DCBE to a VCRP, the upper VCM uses one VCRP and one DCBE.

When issuing a transmit chain request, the LAN driver requires that a certain number of bytes of the transmit frame be able to fit in the VCRP following the VCRP$T_DATA field. The upper VCM does not need to place this number of bytes in the VCRP, but the upper VCM must make sure that there is space for these bytes. The minimum number of bytes required to be able to fit in the VCRP varies across the different LAN devices. The LAN driver stores the minimum space required in the field VCIB$W_DLL_CHAIN_SIZE.

The following list describes the steps required to set up the fields for a chained request. Other fields are also required; but these are the fields that are different for chained and unchained transmit requests.

- The VCRP that the upper VCM allocates for a chained transmit request cannot be smaller than VCRP$T_DATA plus the contents of VCIB$W_DLL_CHAIN_SIZE. When the upper VCM calculates the size required for the LAN header and any of its data, it makes sure that size is not smaller than VCRP$T_DATA plus the contents of VCIB$W_DLL_CHAIN_SIZE.
- Some of the packet is placed into the VCRP. Usually just the headers are placed in the VCRP; specifically the LAN header and the upper VCM's header. The upper VCM's header begins at VCRP$T_DATA + header size (where header size is either LAN$C_MAX_HDR_SIZE or the contents of VCIB$W_DLL_HDR_SIZE). The upper VCM copies its header into the VCRP. If it has the LAN header (from a previous call to BUILD_HDR), then the upper VCM can copy the LAN header into the VCRP; placing it just before upper VCM's header.

  Note that if LAN$C_MAX_HDR_SIZE is not defined, then the constant 58 (decimal) can be used.
- Set VCRP$L_BOFF to be the number of bytes to either the LAN header (if the LAN header has been copied into the VCRP) or to the upper VCM's header.
- Set VCRP$L_BCNT to be equal to the number of bytes that have been placed in the VCRP; including the LAN header if it has been copied into the VCRP.
- Set VCRP$A_DCB_LINK to point to the DCBE.
- Set either DCBE$L_BUFFER_ADDRESS or both DCBE$L_SVAPTE and DCBE$L_BOFF to point to the data. If DCBE$L_SVAPTE and DCBE$L_BOFF point to the data, then DCBE$L_BUFFER_ADDRESS is set to zero.
- Set DCBE$L_BCNT to be equal to the number of bytes pointed to by the DCBE.
- Set VCRP$L_TOTAL_PDU_SIZE to the value in VCRP$L_BCNT plus the value in DCBE$L_BCNT.
- Set DCBE$A_DCB_LINK to zero.

Table 6–1 describes the fields that are used for each of these transmit buffer formats.

Table 6–1: Transmit request fields

Field Name	Unchain	Chain	Destroyed
VCRP$L_BOFF	UI	UI	Y
VCRP$L_BCNT	UI	UI	Y
DCBE$L_BOFF	NA	UO	N
DCBE$L_BCNT	NA	UI	N
DCBE$L_BUFFER_ADDRESS	NA	UO	N
DCBE$L_SVAPTE	NA	UO	N

Key to Unchain and Chain

UI - Initialized by the upper VCM
UO - Optionally initialized by the upper VCM
NU - Not used by the LAN driver
NA - Not applicable to this buffer format

Key to Destroyed

Y - Destroyed by the LAN driver
N - Not destroyed by the LAN driver

For the chained format, the upper VCM either initializes the BOFF and SVAPTE fields in the DCBE or initializes the BUFFER_ADDRESS field in the DCBE. If the SVAPTE and BOFF fields are used, then the BUFFER_ADDRESS field is set to zero. If the BUFFER_ADDRESS field is used, then the SVAPTE and BOFF fields are set to zero.

6.5 RECEIVING A PACKET

When the LAN driver calls the upper VCM's RECEIVE_COMPLETE service routine, the VCRP contains the following information.

The field VCRP$A_LAN_R_HEADER contains the address of the destination address field of the LAN header of the frame received. The upper VCM can examine the header for the destination address and the source address. The destination address is always the first 6 bytes of the header and the source address is always the next 6 bytes of the header.

The LAN driver only calls the upper VCM's RECEIVE_COMPLETE service routine when the VCI port is enabled. If the port has not been enabled or if the port has been deemed unusable (via the PORT_UNUSABLE event), then the LAN driver does not pass receive VCRPs to the upper VCM.

The user data of the packet is always embedded within the VCRP. The BOFF and BCNT fields are initialized by the LAN driver to point to the data embedded in the VCRP.

Table 6–2 describes the fields that are used for receive VCRPs.

Table 6–2: Receive complete fields

Field Name	VCRP	Destroyed
VCRP$L_BOFF	LI	Y
VCRP$L_BCNT	LI	Y

Key to VCRP

LI - Initialized by the LAN driver

Key to Destroyed

Y - Destroyed by the upper VCM
N - Not destroyed by the upper VCM

6.6 GETTING AN EVENT

The upper VCM's report event routine is called by the LAN driver for different reasons. This section describes what the upper VCM should do for some of the events.

6.6.1 PORT USABLE AND PORT UNUSABLE

The upper VCM is called with the port usable event as soon as the VCI port can be used for data transfer.

The upper VCM is called with the port unusable event when the VCI port can no longer be used for data transfer. If the upper VCM disabled the VCI port itself, then the port unusable event was expected. If the upper VCM did not disable the port, then the port unusable event is unexpected. In this case, the upper VCM should stop all transmit requests until the VCI port becomes usable.

If the upper VCM enabled automatic restart (NMA$C_PCLI_RES), then the LAN driver attempts to restart the VCI port. In this case, the upper VCM can wait for the LAN driver to restart the VCI port. If automatic restart is not enabled, then the upper VCM has to start the port itself using an ENABLE_PORT request.

6.6.2 NEW ADDRESS

When the new address event is reported to the upper VCM, it will have to ask the LAN driver to rebuild all the LAN headers that the upper VCM has stored from the BUILD_HDR service routine. This is because those headers contain the old address and are incorrect. If the upper VCM has not used the BUILD_HDR routine, then it can ignore this event.

6.6.3 RESTART FAILED

This event is reported to the upper VCM when the LAN driver has decided to leave the VCI port in a disabled state. This happens when the LAN driver has failed to restart the VCI port thirty consecutive times. This event is only reported to upper VCMs that have automatic restart enabled.

Appendix A

VCI User IDs

Note that the base used for the VCI ID is hexadecimal. The VCI IDs for the VCMs are defined by the constants VCI$K_ID_xxx, where xxx is the acronym listed in the table below. These VCI IDs are defined in $VCIBDEF in LIB.MLB.

Table A–1: Registered VCI users

VCM Name	VCI Id	Acronym	DESCRIPTION
Network Management	0800	NWM	The Digital Network Architecture Network Management
Network Management	0801	CONF	Conformance Test Tool
Internal use	07xx		Digital internal use
Session Control	0500	SCL	The Digital Network Architecture Session Control Layer
Session Control	0501	SCLSRV	Session Control Session Services
Session Control	0502	SCLMIN	Session Control Minimum Services
Transport - NSP	0400	NSPTP	The Digital Network Architecture proprietary Transport protocol
Transport - OSI	0401	OSITP	The Digital Network Architecture OSI Transport protocol
Transport - Local	0402	LCLTP	Local Transport
Transport - SCA	0403	SCATP	The VMS System Communication Architecture
Transport - LAT	0404	LAT	The Digital Network Architecture Local Area Transport protocol - LAT
Transport - LAST	0405	LAST	The Local Area System Transport protocol - LAST
Transport - LAVC	0406	LAVC	The Local Area VAX-Cluster Transport protocol - LAVC
Transport - MOP	0407	MOP	Maintenance Operation Protocol
Transport - IP	0408	IP	Internet Protocol

Table A–1 (Cont.): Registered VCI users

VCM Name	VCI Id	Acronym	DESCRIPTION
Transport - AMDS	0409	AMDS	Availability Manager
Transport - Customer	0499	CUSTP	Customer Transport Layer
Routing - Network	0300	NRL	Digital Network Architecture Network Routing
Routing - Alias	0301	ALIAS	Cluster Alias Routing
Routing - Customer	0399	CUSRL	Customer Routing Layer
Data Link - LAN	0201	LAN	802.3, Ethernet, FDDI, and Token Ring
Data Link - DDCMP	0202	DDCMP	DDCMP
Data Link - HDLC	0203	HDLC	HDLC
Data Link - ASY	0204	ASY	Asynchronous
Data Link - X25	0205	X25	X.25
Data Link - ADM	0206	ADM	ALTSTART DDCMP Module
Data Link - ACM	0207	ACM	ALTSTART CSMACD Module
Data Link - LAPB	0208	LAPB	LAPB - synchronous
Data Link - LLC2	0209	LLC2	LLC2
Data Link - NETBEUI	0210	NETBEUI	Microsoft LAN protocol
Data Link - IPX	0211	IPX	NETware IPX protocol
Data Link - DSP	0212	DSP	Digital stream protocol
Data Link - Customer	0299	CUSDL	Customer Data Link Layer

Appendix B

Revision History

Table B–1: Revision history

Version	Date	Changes
1.0	1-Apr-1993	First release