NETBIOS Application Development Guide

68X2270
S68X-2270-00

First Edition (April 1987)

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

It is possible that this publication may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

THE PUBLICATION OF THE INFORMATION CONTAINED HEREIN IS NOT INTENDED TO AND DOES NOT CONVEY ANY RIGHTS OR LICENSES, EXPRESS OR IMPLIED, UNDER ANY IBM PATENTS, COPYRIGHTS, TRADEMARKS, MASK WORKS, OR ANY OTHER INTELLECTUAL PROPERTY RIGHTS.

Requests for copies of this publication and for technical information about IBM products should be made to your IBM Authorized Dealer or your IBM Marketing Representative.

Preface

This manual describes the software interface of the Network Basic Input Output System (NETBIOS) and the specific implementation of NETBIOS used with the IBM PC Network and the IBM PC Network Protocol Driver.

The information in this manual is intended for developers who provide software products to operate with the NETBIOS.

This manual is divided into the following sections.

"Section 1. NETBIOS" discusses the NETBIOS and Network Control Blocks (NCBs). It also provides an NCB command overview and a reference that describes the NCB return codes.

"Section 2. PC Network Adapter" describes the implementation of the NETBIOS by the PC Network Adapter. It also includes the pseudocode for the NETBIOS commands and the frame types for the PC Network Adapter.

"Section 3. PC Network Protocol Driver" describes the implementation of the NETBIOS by PC Network Protocol Driver. It also includes the pseudocode for the NETBIOS commands and the frame types for the PC Network Protocol Driver.

A Glossary and an Index are also provided.

Suggested Related Publications

- *Macro Assembler for the IBM Personal Computer*
- *IBM Token-Ring Network NETBIOS Program Formats and Protocols*
- *IBM Token-Ring Network Adapter Technical Reference*
- *IBM PC Network Adapter II Technical Reference*
- *IBM PC Network Baseband Adapter Technical Reference*
- *IBM PC Network Adapter II/A Technical Reference*
- *IBM PC Network Baseband Adapter II Technical Reference*
- *IBM PC Network Adapter Technical Reference*

For more information, contact your place of purchase or an IBM Authorized Dealer.

Contents

Figures

Section 1. NETBIOS

Introduction

The Network Basic Input/Output System (NETBIOS) is a software interface that allows computers to communicate over a local area network. The NETBIOS provides four major groups of support functions:

- Session Support

  The NETBIOS creates a session and allows interchange of information with another user (name) on the network. This is a point-to-point connection.

- Datagram Support

  Datagram support allows messages to be sent to a name, group name, or to the entire network and to be received from a name or group name. They do not provide a point-to-point connection and there is no guarantee of message transfer or reception.

- Name Support

  The NETBIOS supports name control in two ways on the network.

  - Each node has a permanent node name.
  - Multiple users within a node are defined with user-assignable names for network configuration. This is done under operator or application control. A name is given to represent each user. The names can be general names, such as "John," instead of specialized names or numbers. These names can also be clustered into logical groups.

- Network Adapter Status and Control Support

NETBIOS also provides a structured programming interface using Network Control Blocks (NCBs).

Network Program Control

The NETBIOS provides program control of the network and presents the interface to the user program. This interface should be used to program the network adapter. The NETBIOS places the unique features of a local area network into a standard format.

It is the responsibility of the operating system or application program to make sure that data or devices are secure on the network as network security is not built into the NETBIOS.

Using the Network

To transfer data using session support:

1. Add your name to the table of names on the adapter. This is the name that you are known by on the network. Skip this step when using the permanent node name.
2. Establish a session with another name on the network. This gives you a logical connection with another name. The other name can be in your name table or in a name table of another adapter.
3. Send and receive messages using that session.

As an alternative to session support, you can use datagram support to transfer data on the network.

Network Performance

To optimize the performance of each adapter:

- Send data in large blocks whenever possible. This reduces network overhead and maximizes throughput.
- Use session support, **not** datagram support. This allows larger and more reliable data transfer.
- Issue multiple commands. Issuing multiple receive commands allows overlap of network activity and local input/output (I/O) on the computer. Multiple transmissions of large data blocks on one adapter and multiple receptions on the other end maximizes throughput in a single point-to-point session.

- Configure the adapter to only the number of sessions and commands you need. Use the Reset command to configure your NETBIOS to the smallest number of sessions and outstanding commands, because the resources are limited. The fewer the configured sessions, the larger the frame transmitted on the network. After space is set aside for session entries and command blocks, the remaining space is used for buffers. The size of the frame transmitted on the network is equal to or less than the size of the transmission buffer.

Network Control Block (NCB)

The following describes how to design your program with the NETBIOS by using a collection of fields to form a Network Control Block (NCB). Once a command is presented to the NETBIOS in the NCB format, a response is returned to the program in the form of a return code inside the NCB.

Note: The following discussion describes the NCB commands and return codes that are common to all implementations of the NETBIOS. For information about commands or return codes that are not described below, refer to the specific implementation of the NETBIOS that you are using.

NCB commands provide control of the adapter on the network. The commands are divided into four categories:

- General
- Name support
- Session support
- Datagram support.

Note: The following discussion assumes you are familiar with assembler language and its concepts.

Commands are presented to the NETBIOS in the form of an NCB. The following is the basic concept and format to present NCB commands to the NETBIOS:

1. Build and fill in all required fields of an NCB.

   When you build a new NCB, set all fields to binary zeros. (See "NCB Field Description" on page 1-10 for the required fill character for each field.)

2. Allocate any necessary buffers specified in the required NCB fields.

3. Make sure at least 20 bytes of stack space are left for each outstanding NCB command.

4. Place the address of the NCB in the ES:BX register pair. Issue software interrupt hex 5C.

5. Once the interrupt is issued, *do not* change or move it until the NCB command is processed.
6. After the command is processed, control is returned to the caller. The result of the process is in the AL register, the Return Code field of the NCB, and the Command Complete field of the NCB.

You must either fill in all required parameters or use the default values (by specifying binary zeros in certain fields) for some of the commands in the NCB. Command codes and return codes are represented by hexadecimal values.

All NCB commands, except the Reset and Cancel commands, have wait and no-wait options. The wait option means that when you issue the command, the microprocessor waits until the command executes before returning to the next instruction. The no-wait option means that the microprocessor returns immediately after processing the command and is interrupted at the post address when the command executes.

Wait Option

When you use the wait option, check either the NCB_RETCODE field, the CMD_CPLT field, or the AL register for the return code. See page 2-9 for multitasking considerations.

No-Wait Option

When the no-wait option is used, the issued command returns to the next instruction after the return from interrupt hex 5C. The AL register should be checked for hex 00 (command complete). Refer to the description of each command issued for any other values returned.

If the command was accepted with AL equal to hex 00, an interrupt occurs when the command is processed. Either the AL register, the NCB_RETCODE, or the CMD_CPLT field may be checked for the return value. If the command is accepted but not complete, both the NCB_RETCODE field and the CMD_CPLT field should contain hex FF. Be sure to specify each NCB_COMMAND field correctly or you will receive a return code of hex 03, invalid command.

NCB Format

The following figure shows the format of the NCB fields.

Note: An "@" is used to represent the word "address" throughout this document.

Field Name	Length (Bytes)	8086 Type	Description
NCB_COMMAND	1	DB	NCB Command field
NCB_RETCODE	1	DB	NCB Return Code field
NCB_LSN	1	DB	NCB Local Session Number field
NCB_NUM	1	DB	NCB number assigned to your name
NCB_BUFFER@	4	DD	NCB pointer to message buffer address (offset:segment)
NCB_LENGTH	2	DW	NCB buffer length (in bytes)
NCB_CALLNAME	16	DB	NCB name on local or remote adapter. For CHAIN SEND the first 2 bytes indicate the length of the second buffer and the next 4 bytes point to the second buffer address.
NCB_NAME	16	DB	NCB name on local adapter
NCB_RTO	1	DB	NCB session receive time-out value
NCB_STO	1	DB	NCB session send time-out value
NCB_POST@	4	DD	NCB pointer to POST routine. (offset:segment)
NCB_LANA_NUM	1	DB	Use hex 00 for the primary adapter. Use hex 01 for the alternate adapter.
NCB_CMD_CPLT	1	DB	NCB Command Complete Status field
NCB_RESERVE	14	DB	NCB reserved area

Figure 1-1. Network Control Block Format

The following figure summarizes the NCB error codes that are returned by the adapter.

No Wait Commands

Error Code	No Post Routine	Post Routine
Immediate Error Code	AL NCB_RETCODE NCB_CMD_CPLT * see 1	AL NCB_RETCODE NCB_CMD_CPLT * see 1
Final Error Code	If no immediate error, NCB returns: AL = hex 00, NCB_CMD_CPLT = hex FF NCB_RETCODE = hex FF (pending) * see 2	AL NCB_RETCODE NCB_CMD_CPLT * see 3

Wait Commands

Error Code	No Post Routine
Immediate Error Code	AL NCB_RETCODE NCB_CMD_CPLT *see 1
Final Error Code	AL NCB_RETCODE NCB_CMD_CPLT

1. If the immediate error is not zero, it is also the final error. The adapter ends processing if an immediate error occurs.
2. Poll on NCB_CMD_CPLT when it changes from hex FF. NCB_CMD_CPLT, NCB_RETCODE, and AL will contain the final error or hex 00 (success).
3. Use the NCB_CMD_CPLT field of the NCB only to return the error when there is no POST address. For example: (NCB_POST@ = 0000:0000).

Figure 1-2. NCB Error Code Summary

NCB Field Description

The following is a list of the NCB fields and the description of each field.

NCB_COMMAND

NCB_COMMAND is a 1-byte field. Each command executes in either a wait or no-wait mode. Setting the high-order bit to 1 selects the no-wait option. Setting the high-order bit to 0 selects the wait option. The remaining 7 bits specify the command that you want the adapter to execute.

Use the no-wait option for maximum throughput between the adapter and the computer. In addition, the no-wait option allows multiple commands to be queued for execution within the adapter.

The programming interface to the NETBIOS is different depending upon the selection of the wait/no-wait option. For example; when issuing the instruction interrupt hex 5C using the wait option, control is not returned to the next instruction until the adapter completes the command. When the command is complete, check the AL register, the NCB_RETCODE field, or the CMD_CPLT field for the status of the completed command.

If you choose the no-wait option for the interrupt hex 5C instruction, you will receive two return codes. One code is returned immediately after you issue the instruction. The following is a list of the possible immediate return codes found in the AL register:

Immediate Return Codes

- Hex 00—Command has no immediate errors
- Hex 03—Invalid command
- Hex 21—Interface busy
- Hex 22—Too many commands outstanding
- Hex 23—Invalid number in NCB_LANA_NUM field
- Hex 24—Command completed while cancel occurring
- Hex 26—Command not valid to cancel
- Hex (40 – 4F)—Unusual network condition
- Hex (50 – FE)—Adapter malfunction

When the immediate return code is hex 00, a final return code is posted to the user after the command is executed. The return code is posted by interrupting the user application and can be noted by checking the NCB_CMD_CPLT field. If the NCB_POST@ field is not zero, the adapter interrupts the user application program at the address specified in the NCB_POST@ field. If the NCB_POST@ field is zero, the adapter does not interrupt the program, and command completion must be determined by checking the NCB_CMD_CPLT field.

When the adapter interrupts the user application program upon command completion, the final return code can be obtained from either the AL register, the NCB_RETCODE field, or the NCB_CMD_CPLT field. If checking the NCB_CMD_CPLT field, a change in value from hex FF (pending status) indicates command completion. This value represents the final return code. The possible final return codes vary for each command.

If the immediate return code is other than hex 00, the adapter cannot execute the requested command and adapter processing ends. See "NCB Return Code Causes and Actions" on page 1-60.

NCB_RETCODE

NCB_RETCODE is a 1-byte field indicating the return code of a command. If the return code is hex 00, the operation was successful. Any other number indicates the operation failed or has not completed. If the no-wait option is used without being interrupted on command completion (with POST@ =00), the NCB_CMD_CPLT field, not the NCB_RETCODE field, should be used to determine when the command completes. Until the command executes, NCB_CMD_CPLT will contain hex FF. See "NCB Return Code Causes and Actions" on page 1-60.

When a command has completed, the main program is interrupted by the adapter at the POST address. You can choose to check the AL register or the NCB_CMD_CPLT field instead of the NCB_RETCODE field.

Loop on the NCB_CMD_CPLT field instead of going into a program loop on the NCB_RETCODE field when waiting for a command to complete its execution.

NCB_LSN

NCB_LSN is a 1-byte field indicating the local session number. This is the number of the session you have with another name on the network. This is only valid after a Call or Listen command has successfully executed. For Send and Receive commands under session support, this field must be correctly filled in. For datagram support, NCB_LSN does not apply.

A number ranging from 1 to 254 is assigned to the NCB_LSN field using a round-robin modulo 255 technique. Hex 00 and hex FF are never returned.

The Reset command uses this field to indicate the maximum number of sessions supported.

NCB_NUM

NCB_NUM is a 1-byte field indicating the number returned to you after an Add Name or Add Group Name command is executed. This number, not the name, must be used with all datagram support commands and any Receive Any commands.

The NCB_NUM field is assigned a number in a round-robin modulo 255 technique ranging from 2 to 254. The permanent node name number is always 1. Hex 00 and hex FF are never returned.

The Reset command uses this field to indicate the maximum number of command blocks to be supported.

NCB_BUFFER@

NCB_BUFFER@ is a 4-byte field indicating a pointer to the buffer you wish to use with a command. This field is in Define Double-Word format (offset:segment) and must be a valid address in memory. See the *IBM Macro Assembler* manual for references to define double-word format (DD).

NCB_LENGTH

NCB_LENGTH is a 2-byte field indicating the length of the data, in bytes, you want transferred.

For a Receive, Receive Any, Status, Session Status, Receive Broadcast Datagram, and Receive Datagram command, this field is used and updated to indicate the number of bytes that are actually received. For a Send, Chain Send, Send Datagram, and Send Broadcast Datagram command, this field is used to indicate the number of bytes to be sent.

NCB_CALLNAME

NCB_CALLNAME is a 16-byte field indicating the name with whom you want to communicate. All 16 bytes are used. The name can be either on your adapter or any other adapter.

For a Chain Send command, the first 6 bytes are used to specify the second buffer. In these 6 bytes, the first 2 bytes specify the length field and the last 4 bytes specify the buffer address. They are in the same format as the NCB_LENGTH and NCB_BUFFER@ fields.

NCB_NAME

NCB_NAME is a 16-byte field indicating the name that you are known by on the network. All 16 bytes must always be used. The table on the adapter can hold up to 16 names. You are always known by the permanent node name on the network. The permanent node name is 10 bytes of binary zeros, followed by the first 6 bytes of the node ID returned by the Status command.

NCB_RTO

NCB_RTO is a 1-byte field used by the Call and Listen commands to specify a time-out period for all Receive commands associated with that session. The receive time-out is the maximum amount of time allowed before a Receive command returns a time-out error. The time-out value is specified in increments of 500 ms. If binary zero (hex 00) is specified, there is no time-out. The time-outs can also be different for each session, but are fixed once the session is established.

NCB_STO

NCB_STO is a 1-byte field used by the Call and Listen commands to signify a send time-out. The send time-out is the maximum amount of time allowed before a Send command returns a time-out error. The

time-out is specified in increments of 500 ms. SEND time-outs should be used with caution because they will always drop the session if they occur. If binary zero (hex 00) is specified, there is no time-out.

NCB_POST@

NCB_POST@ is a 4-byte field indicating the address of the routine that is to be executed when the adapter has completed processing a command. This field is used in no-wait options only. This field is in Define Double-Word format (offset:segment) and must be a valid address in memory. Your POST routine must establish DS and any other registers needed. Only AL, CS, ES, and BX registers are set for the NCB being completed. The POST routine is called by the adapter on interrupt level with interrupts masked. To return, issue an IRET. Your POST routine should be short and return immediately, unless you unmask interrupts.

If the POST address is specified as all zeros, a POST does not occur. This allows a program to do other work and then loop until the NCB_CMD_CPLT field changes from hex FF. When the change from hex FF occurs, either the command executed or an error code was returned. Do not check the NCB_RETCODE because it changes before the command is completed. This can be useful in a BASIC program by using PEEK and POKE.

NCB_LANA_NUM

NCB_LANA_NUM is a 1-byte field indicating which adapter you want to use. A value of hex 00 directs the command block to the primary adapter. A value of hex 01 directs the command block to the alternate adapter.

NCB_CMD_CPLT

NCB_CMD_CPLT is a 1-byte field indicating the command status. A value of hex FF in this field indicates the command is pending. A value of hex 00 means that the command was completed. Any other value means that the command executed with an error. See "NCB Return Code Causes and Actions" on page 1-60.

This byte is only useful if the NCB_POST@ was specified as all zeros while using a no-wait option command. Otherwise, check the NCB_RETCODE field.

NCB_RESERVE

NCB_RESERVE is a 14-byte reserved field. This space must be allocated because the NETBIOS uses this field to store temporary variables.

NCB Commands

Network Control Block commands are divided into four support categories. The following figure shows the commands in each support category:

General	Name Support	Session Support	Datagram Support
RESET	ADD NAME	CALL	SEND DATAGRAM
CANCEL	ADD GROUP NAME	LISTEN	SEND BROADCAST DATAGRAM
STATUS (ADAPTER STATUS)	DELETE NAME	HANG UP	RECEIVE DATAGRAM
UNLINK		SEND	RECEIVE BROADCAST DATAGRAM
		CHAIN SEND	
		RECEIVE	
		RECEIVE ANY	
		SESSION STATUS	

Figure 1-3. NCB Support Categories

General Commands

General commands allow the adapter to read status, or to control other outstanding commands on the network.

RESET

Command Code:

Hex 32 —Wait option

Function:

This command resets the local status and clears the name and session tables. The number of sessions and NCB command blocks to be supported can be changed by this command. For the maximum values of sessions and NCB command blocks, refer to the specific implementation you are using. The default values are 6 sessions and 12 command blocks. Additional sessions and NCB command blocks require space and consequently reduce the session frame size on the network. For best performance, configure only the number of sessions and commands that you actually need. If the specified values exceed the limits, the maximum values are used. If binary zeros are specified, the default values are used.

Once the reset is complete, the Status command can be used to see the resulting maximum data frame size allowed by the adapter. Since overall performance is related to the number of frames sent on the network, you can choose to optimize your message size to fit into one frame. If two adapters are reset to different values of NCB command blocks and sessions supported, the resulting frame sent on the network between these two adapters under a session will always be the smaller of the two.

Only a power-on reset will reset the traffic and error statistic data returned from the Status command. The Reset command does not.

Fields Required:

NCB_LSN (number of sessions supported)
NCB_NUM (number of command blocks supported)

NCB_LANA_NUM (number of the adapter to reset)

Fields Returned:

NCB_RETCODE
NCB_CMD_CPLT

Final Return Codes:

Hex 00 —Command complete
Hex 03 —Invalid command
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

CANCEL

Command Code:

Hex 35 —Wait option

Function:

This command requests that the command, whose NCB can be found at the address given by NCB_BUFFER@, be canceled.

The Cancel command return code indicates the status of the Cancel command, or the reason for its failure. For example, a return code of hex 00 means only that the Cancel command was issued. It does not mean the command to be canceled was actually canceled. To determine the status of the command to be canceled, you must check the return code for that command.

The following commands are not valid to CANCEL: ADD NAME, ADD GROUP NAME, DELETE NAME, SEND DATAGRAM, SEND BROADCAST DATAGRAM, SESSION STATUS, RESET, and CANCEL. Use caution when canceling a Send command because the session will be dropped.

Fields Required:

NCB_LANA_NUM (number of the adapter)
NCB_BUFFER@ (address of the NCB to be canceled)

Fields Returned:

NCB_RETCODE
NCB_CMD_CPLT

Final Return Codes:

Hex 00 —Command completed
Hex 03 —Invalid command
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex 24 —Command executed while cancel was occurring
Hex 26 —Command not valid to cancel
Hex (40 − 4F) —Unusual network condition
Hex (50 − FE) —Adapter malfunction

STATUS (ADAPTER STATUS)

Command Code:

Hex 33 —Wait option
Hex B3 —No-wait option

Function:

This command provides status information for a local or remote adapter by the name specified in the NCB_CALLNAME field. If an asterisk (*) is specified in the first byte of the NCB_CALLNAME field, the information for the local adapter is returned. This information is placed in the buffer address specified, NCB_BUFFER@, and the length field is updated to indicate the number of bytes of information received.

The minimum number of bytes in the status buffer is 60. The maximum number of bytes required to hold the status information depends on the maximum number of names that can be supported. For the maximum values, refer to the specific implementation of the NETBIOS that you are using. In general, the number of required bytes is 60 + 18(X), where X is the number of names added to the adapter.

Note: A return code of hex 06 is posted in the NCB_RETCODE field if the receive buffer is not large enough for the data. The remaining data is lost at this point.

Fields Required:

NCB_BUFFER@
NCBI_LENGTH
NCB_CALLNAME (local or remote name, or * for local)
NCB_POST@ (If no-wait option used)
NCB_LANA_NUM (Number of the adapter)

Fields Returned:

NCB_RETCODE
NCB_CMD_CPLT
NCB_LENGTH

Immediate Return Codes:

Hex 00 —Command has no immediate errors
Hex 03 —Invalid command
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

Final Return Codes:

Hex 00 —Command complete
Hex 01 —Invalid buffer length
Hex 03 —Invalid command
Hex 05 —Command timed out
Hex 06 —Message incomplete
Hex 0B —Command canceled
Hex 15 —Illegal name
Hex 19 —Name conflict detected
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

Note: For information about data that is returned in the Data Buffer field refer to the specific implementation of the NETBIOS that you are using.

UNLINK

Command Code:

Hex 70 —Wait option

Function:

This command is used only with the remote program load (RPL) feature. To see if this command is actually supported, refer to the specific NETBIOS implementation that you are using. The command applies only if a call to IBMNETBOOT was made at power-up time of this computer. The session with IBMNETBOOT is dropped when this command is issued. The BIOS also ends the INT 13 redirection to the network. For more information on RPL, refer to the technical reference manual for your adapter (not applicable to the IBM PC Network Adapter).

Fields Required:

NCB_LANA_NUM (Number of the adapter you want to unlink.)

Fields Returned:

NCB_RETCODE
NCB_CMD_CPLT

Immediate Return Codes:

Hex 00 —Command has no immediate errors
Hex 03 —Invalid command
Hex 21 —Interface busy
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

Final Return Codes:

Hex 00 —Command complete
Hex 03 —Invalid command
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

Name Support

Name support commands allow the computer to be known by a name on the network. A name can be a unique name or a group name on the network. The adapter checks to see if a name is unique on an Add Name and returns an error if anyone else is using the name you want to add. When Add Group Name is used, the same name can be added by many adapters on the network.

The number of names that each NETBIOS implementation can support varies. For the maximum values, refer to the specific NETBIOS implementation that you are using. A permanent node name is always present and consists of 10 bytes of binary zeros followed by the unique adapter unit ID number. This permanent node name is also unique on the network. You can find its value by issuing a Status NCB for a local status by putting an asterisk (*) in the CALLNAME field. Look at the first 6 bytes returned in the buffer specified. Append this number to 10 bytes of binary zeros to make a total of 16 bytes for the permanent node name. This permanent name does not show up as an entry in the local name table returned by the NCB command, STATUS.

The Reset command deletes all names from the specified adapter with the exception of the permanent node name.

Reserved Names

Any name starting with an * in ASCII or hex 00 is reserved and cannot be added or deleted.

It is not recommended that you use any name starting with IBM.

ADD NAME

Command Code:

Hex 30 —Wait option
Hex B0 —No-wait option

Function:

This command adds a 16-character name to the table of names. The name added cannot be added by anyone else on the network.

When the NETBIOS processes this command, it sends broadcast requests on the network. If no reply is received, the name is assumed to be unique and is added to the table of names. The command returns the number of your name in the NCB_NUM field. This number is used in datagram support and for Receive Any commands.

Fields Required:

NCB_NAME
NCB_POST@ (If no-wait option used)
NCB_LANA_NUM (Number of the adapter for the add name.)

Fields Returned:

NCB_RETCODE
NCB_NUM
NCB_CMD_CPLT

Immediate Return Codes:

Hex 00 —Command has no immediate errors
Hex 03 —Invalid command
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

Final Return Codes:

Hex 00 —Command completed
Hex 03 —Invalid command
Hex 09 —No resource available

Hex 0D —Duplicate name in local name table
Hex 0E —Name table is full
Hex 15 —Name invalid
Hex 16 —Name in use on remote node
Hex 19 —Name conflict detected
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

ADD GROUP NAME

Command Code:

Hex 36 —Wait option
Hex B6 —No-wait option

Function:

This command adds a 16-character name to the table of names. Group names allow a single node to communicate with many nodes. The name added cannot be added by anyone else on the network as a unique name but can be added by anyone as a group name.

When the NETBIOS processes this command, it sends a broadcast request on the network. If no unique name replies, the name is added. The command returns the number of the name in the NCB_NUM field. This number is used in datagram support and for Receive Any commands.

Fields Required:

NCB_NAME
NCB_POST@ (If no-wait option used)
NCB_LANA_NUM (Number of the adapter for the add group name.)

Fields Returned:

NCB_RETCODE
NCB_NUM
NCB_CMD_CPLT

Immediate Return Codes:

Hex 00 —Command has no immediate errors
Hex 03 —Invalid command
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

Final Return Codes:

Hex 00 —Command completed
Hex 03 —Invalid command
Hex 09 —No resource available
Hex 0D —Duplicate name in local name table
Hex 0E —Name table is full
Hex 15 —Name invalid
Hex 16 —Name in use on remote node
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

DELETE NAME

Command Code:

Hex 31 —Wait option
Hex B1 —No-wait option

Function:

This command deletes a 16-character name from the name table. Use the Hang Up command before you delete the name. If the name has active sessions when this command is issued, the name is flagged as deregistered and the status, hex 0F (command completed, name has active sessions) is returned to the user. The actual deletion of the name does not occur until all sessions associated with the name are closed or abnormally terminated. If the name has only pending non-active session commands when the Delete Name command is issued, the name is removed and hex 00 (command completed) is returned to the user. The pending non-active session commands are terminated immediately with the "name was deleted" status. Non-active session commands are: Listen, Receive Any, Datagram Receive, and Receive Broadcast Datagram.

A name flagged as deregistered continues to occupy an entry in the local name table until the deregistration is complete.

Fields Required:

NCB_NAME
NCB_POST@ (If no-wait option used)
NCB_LANA_NUM (Number of the adapter for the delete name.)

Field Returned:

NCB_RETCODE
NCB_CMD_CPLT

Immediate Return Codes:

Hex 00 —Command has no immediate errors
Hex 03 —Invalid command
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding

Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

Final Return Codes:

Hex 00 —Command completed
Hex 03 —Invalid command
Hex 0F —Command completed, name has active sessions and is deregistered.
Hex 15 —Name Invalid
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

Session Support

Session support commands allow you to establish a logical connection (session) on the network, send and receive messages, end sessions, and obtain session status. More than one command can be outstanding because the connection is in two-way simultaneous transmission mode.

Sessions are established between any two names on the network. These names can be on your adapter or any other adapter. Names are used to establish sessions, but a 1-byte number is used to refer to each session after they are established. This number is found in the NCB_LSN field that is returned when a session is established. The number of sessions that each NETBIOS implementation can support varies. For the maximum values, refer to the specific NETBIOS implementation that you are using. The same name pair can be used to establish more than one session. The difference between the session pairs is the different LSN fields. If you create a local session, two session entries are used instead of one. One side of the local session has one LSN number associated with it, and the other side has a different local session number (LSN) number. To establish a session with yourself, make the fields, CALLNAME and NAME, equal.

Session support provides end-to-end data transfer of messages up to 65,535 bytes in length. The Reset command immediately ends all sessions.

CALL

Command Code:

Hex 10 —Wait option
Hex 90 —No-wait option

Function:

This command opens a session with another name specified by the NCB_CALLNAME field using the local name specified by the supplied NCB_NAME. The name that you call must have a Listen command pending for the session to be established. You can establish a session with either a local or a remote name. Multiple sessions can be established with the same pair of names. All Send and Receive commands for this session will immediately end if they are unsuccessful after the specified time-out intervals. The time-out intervals are specified in 500-ms units (a value of zero means that no time-out will occur). The Call command is repeated internally a predetermined number of retries within an internal system time-out value. The Call command ends unsuccessfully after the system time-out intervals and retry counts are exhausted. When the Call is complete, a local session number (LSN) is assigned and used thereafter to refer to the established session.

Local session numbers (NCB_LSN) are assigned in a round-robin technique, starting from the next available value within the range of 1 to 254.

Fields Required:

NCB_CALLNAME
NCB_NAME
NCB_RTO (Specified in 500-ms increments. If the field is set to hex 00, no receive time-outs occur.)
NCB_STO (Specified in 500-ms increments. If the field is set to hex 00, no send time-outs occur.)
NCB_POST@ (If no-wait option used.)
NCB_LANA_NUM (Number of the adapter issuing the call.)

Fields Returned:

NCB_RETCODE
NCB_LSN
NCB_CMD_CPLT

Immediate Return Codes:

Hex 00 —Command has no immediate errors
Hex 03 —Invalid command
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

Final Return Codes:

Hex 00 —Command completed
Hex 03 —Invalid command
Hex 09 —No resource available
Hex 0B —Command canceled
Hex 11 —Local session table full
Hex 12 —Session open rejected
Hex 14 —Cannot find name called, or no answer, or hex 00
Hex 15 —Invalid name
Hex 18 —Session ended abnormally
Hex 19 —Name conflict detected
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

LISTEN

Command Code:

Hex 11 —Wait option

Note: Use this carefully because it does not time out and your program does not continue until the command is satisfied.

Hex 91 —No-wait option

Function:

This command enables a session to be established with the name specified in the field, NCB_CALLNAME, using the name specified by the field, NCB_NAME. If the field, CALLNAME, has a name starting with * , a session is established with any network node that issues a Call command to the local name specified by the NCB_NAME field.

A Listen command for a specific name has priority over a Listen command for any name. Sessions can be established with either a local or a remote name. Multiple sessions can be established with the same pair of names.

All Send and Receive commands for this session immediately end if they are unsuccessful after the specified time-out intervals. If a Send command times out, the session is abnormally terminated.

The time-out intervals are specified in 500-ms units (a value of zero means that no time-out will occur). A Listen command does not time out, but it occupies a session entry and is considered a pending session in information returned on a Status command. Local session numbers (LSNs) are assigned in a round-robin technique starting with the next available value within the range of 1 to 254. Also, if an * is used for the called name, the name initiating the call will be returned in the field, CALLNAME.

Hex 19 (name conflict detected) is returned if, during the completion of a Listen command, a unique name exists in more than one table. All nodes with the name registered,

except for the one where the Listen command has returned successfully, will report the same error.

Fields Required:

NCB_CALLNAME (This can be specified in the first byte as * . The * is used to listen for a call from anyone to your name. If a name is specified in this field, it takes priority over a name of * .)
NCB_NAME
NCB_RTO (Specified in 500 ms increments. If the field is set to hex 00, no receive time-out occurs.)
NCB_STO (Specified in 500 ms increments. If the field is set to hex 00, no send time-out occurs.)
NCB_POST@ (If no-wait option used)
NCB_LANA_NUM (Number of the adapter you want to listen.)

Fields Returned:

NCB_RETCODE
NCB_LSN
NCB_CALLNAME (* specifies listen to anyone.)
NCB_CMD_CPLT

Immediate Return Codes:

Hex 00 —Command has no immediate errors
Hex 03 —Invalid command
Hex 09 —No resource available
Hex 15 —Invalid name
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

Final Return Codes:

Hex 00 —Command completed
Hex 03 —Invalid command
Hex 0B —Command canceled
Hex 11 —Local session table full
Hex 15 —Invalid name
Hex 17 —Name was deleted
Hex 18 —Session ended abnormally
Hex 19 —Name conflict detected
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

HANG UP

Command Code:

Hex 12 —Wait option
Hex 92 —No-wait option

Function:

This command closes the session with the name on the network indicated by the local session number. Hex 00 (command completed) is returned on a normal close and hex 0A (session closed) or hex 08 (invalid local session number) is returned if the session is already closed or does not exist.

When a Hang Up command is received, all pending local Receive commands are ended and returned to the issuer with "session closed" in the NCB_RETCODE field. The termination is valid regardless of whether any data had been transferred by the pending command. If a local Send command is pending, the Hang Up command delays until the Send is complete or until approximately 20 seconds have elapsed. This delay occurs whether the command has begun to transfer data or is waiting for the remote computer to issue a Receive command. The Hang Up command is performed if any of the following conditions occur:

- The Send command is complete.
- The Send command has immediately ended.
- The Send command fails because the session was terminated by the other computer with a Hang Up command.
- The Send command fails because of the time-out specified when the session was opened.

If one of the above conditions does not occur within 20 seconds after the Hang Up command is executed, the Hang Up command is returned with a hex 05 (command timed out), and the session immediately ends.

When a session closes, all Send and Receive, except Receive Any, commands pending on the closed session are returned to the issuer with hex 0A (session closed). If one Receive Any command is pending for that session, it is returned with a hex 0A (session closed). Only one Receive Any command is returned even though many Receive Any commands may be pending. Even though a single Receive Any command is returned, many Send or Receive commands can be returned when pending.

When a session abnormally ends, all outstanding commands for that session are returned to the issuer with hex 18 (session ended abnormally).

Fields Required:

NCB_LSN
NCB_POST@ (If no-wait option used)
NCB_LANA_NUM (Number of the adapter you want to hang up.)

Field Returned:

NCB_RETCODE
NCB_CMD_CPLT

Immediate Return Codes:

Hex 00 —Command has no immediate errors
Hex 03 —Invalid command
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

Final Return Codes:

Hex 00 —Command completed
Hex 03 —Invalid command
Hex 05 —Command time-out
Hex 08 —Invalid local session number
Hex 0A —Session closed
Hex 0B —Command canceled
Hex 18 —Session ended abnormally
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

SEND

Command Code:

Hex 14 —Wait option
Hex 94 —No-wait option

Function:

This command sends data using the session number indicated in the local session number (LSN). The data is taken from the buffer, NCB_BUFFER @, for the number of bytes, NCB_LENGTH. The buffer size may be up to 65,535 bytes in length.

When a session is closed by the remote computer, all Send commands pending on the closed session are returned with a "session closed" status. If a local Hang Up command is issued with any pending Send commands, the Hang Up command is delayed until the Send commands are completed or ended by error.

If a session ends abnormally, a "session ended abnormally" status is returned. If the SEND time-out expires, the session immediately ends and a "command timed out" status is returned. Time-out values for the Send command are associated with the session when CALL or LISTEN was issued and cannot be specified here.

If more than one SEND is pending, the data is transmitted in a first-in-first-out (FIFO) order within a session.

If the Send command cannot be completed for any reason, the session will be terminated abnormally to guarantee data integrity.

Send commands without corresponding Receive commands consume resources on the adapter. It is not advisable to issue Send commands without corresponding Receive commands.

Fields Required:

NCB_LSN
NCB_BUFFER@
NCB_LENGTH
NCB_POST@ (If no-wait option used)
NCB_LANA_NUM (Number of the adapter you want to send.)

Field Returned:

NCB_RETCODE
NCB_CMD_CPLT

Immediate Return Codes:

Hex 00 —Command has no immediate errors
Hex 03 —Invalid command
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

Final Return Codes:

Hex 00 —Command completed
Hex 03 —Invalid command
Hex 05 —Command timed out
Hex 08 —Invalid local session number
Hex 0A —Session closed
Hex 0B —Command canceled
Hex 18 —Session ended abnormally
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

CHAIN SEND

Command Code:

Hex 17 —Wait option
Hex 97 —No-wait option

Function:

This command sends data by the session number indicated in the LSN. The data is taken from the buffers for the indicated number of bytes. Two buffers can be chained together with this command.

The data in the second buffer is chained to the data in the first buffer and sent as a single message. The NCB_CALLNAME is used to specify the length and address of the second buffer. The length is specified in the first 2 bytes; the buffer address is the next 4 bytes.

When a session is closed by the remote computer, all Chain Send commands pending on the closed session will be returned with a session closed status. If a local Hang Up command is issued with any pending Chain Send commands, the Hang Up execution is delayed until the Send commands are complete.

If a session ends abnormally, a "session ended abnormally" status is returned. If the CHAIN SEND time-out expires, the session immediately ends and a "command timed out" status is returned. Time-out values for the Send command are associated with the session when CALL or LISTEN is issued.

Message lengths can be from 0 to 65,535 bytes in length for each of the two buffers. The maximum number of bytes that can be sent with one Chain Send command is 131,070.

If more than one Chain Send command is pending, the data is transmitted in a first-in-first-out (FIFO) order.

Fields Required:

NCB_LSN
NCB_BUFFER@
NCB_LENGTH
NCB_CALLNAME
The format for the second buffer is specified as follows:
NCB_LENGTH2 DW
NCB_BUFFER2@ DD
NCB_POST@ (If no-wait option used)
NCB_LANA_NUM (Number of the adapter you want to chain send.)

Field Returned:

NCB_RETCODE
NCB_CMD_CPLT

Immediate Return Codes:

Hex 00 —Command has no immediate errors
Hex 03 —Invalid command
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

Final Return Codes:

Hex 00 —Command completed
Hex 03 —Invalid command
Hex 05 —Command timed out
Hex 08 —Invalid local session number
Hex 0A —Session closed
Hex 0B —Command canceled
Hex 18 —Session ended abnormally
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

RECEIVE

Command Code:

Hex 15 —Wait option
Hex 95 —No-wait option

Function:

This command receives data from the specified session. If more than one Receive command is outstanding, they are posted according to the following hierarchy: Receive, Receive Any for a specified name, and Receive Any for any name. Once the commands are sorted according to hierarchy, all of the Receive commands are processed in a first-in-first-out order. Time-out values for RECEIVE are specified during a Call or Listen command and cannot be specified during this command.

When a session is closed by either the local session Close command or by the remote adapter closing the session, all pending NCBs for that session are returned with a session ciosed status.

Note: A return code of hex 06 is posted in the NCB_RETCODE field if the receive buffer is not large enough for the message being sent. You can issue another Receive command to obtain the rest of the information before a time-out occurs.

Fields Required:

NCB_LSN
NCB_BUFFER@
NCB_LENGTH
NCB_POST@ (If no-wait option used)
NCB_LANA_NUM (Number of the adapter you want to receive.)

Fields Returned:

NCB_RETCODE
NCB_LENGTH
NCB_CMD_CPLT

Immediate Return Codes:

Hex 00 —Command has no immediate errors
Hex 03 —Invalid command
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

Final Return Codes:

Hex 00 —Command completed
Hex 03 —Invalid command
Hex 05 —Command timed out
Hex 06 —Message incomplete
Hex 08 —Invalid local session number
Hex 0A —Session closed
Hex 0B —Command canceled
Hex 18 —Session ended abnormally
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

RECEIVE ANY

Command Code:

Hex 16 —Wait option (Use this carefully because it does not time out.)
Hex 96 —No-wait option

Function:

This command receives data from anyone with whom you have a session. You must use your name number instead of your name when issuing this command. Your name number (NCB_NUM) was returned when you issued the Add Name or Add Group Name command. If you specify hex FF in the NCB_NUM field, you receive from anyone with whom you have a session for any of your names. If more than one Receive command is outstanding, they are completed in a first-in-first-out (FIFO) order according to the following hierarchy: Receive, Receive Any for a specified name, and Receive Any for any name.

If a session is closed by the local or remote node, or if the session ends abnormally, only one Receive Any command for that session will have a final code returned regardless of the number pending. If a Receive Any command to a name is pending, it is posted as "session closed" with the LSN field indicating the session that closed. A Receive Any command with no name specified is posted only if no Receive Any command to a name is pending for that session name.

Note: A return code of hex 06 is returned in the NCB_RETCODE field if the receive buffer is not large enough for the message being sent. You can issue another Receive command to obtain the rest of the information.

Application programs should not use a Receive Any command to any name unless they wish to receive messages intended for other programs running in the system.

Fields Required:

NCB_BUFFER@
NCB_LENGTH
NCB_NUM (If this field is hex FF, you receive from any remote name with whom there is a session.)
NCB_POST@ (If no-wait option used)
NCB_LANA_NUM (Number of the adapter you want to receive for any name.)

Fields Returned:

NCB_LSN
NCB_RETCODE
NCB_NUM (If hex FF is specified)
NCB_LENGTH
NCB_CMD_CPLT

Immediate Return Codes:

Hex 00 —Command has no immediate errors
Hex 03 —Invalid command
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

Final Return Codes:

Hex 00 —Command completed
Hex 03 —Invalid command
Hex 06 —Message incomplete
Hex 0A —Session closed
Hex 0B —Command canceled
Hex 13 —Invalid name number
Hex 17 —Name deleted
Hex 18 —Session ended abnormally
Hex 19 —Name conflict detected
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
(50 – FE)H—Adapter malfunction

SESSION STATUS

Command Code:

Hex 34 —Wait option
Hex B4 —No-wait option

Function:

This command receives the status of all active sessions for the specified name. This command optionally gives the status for all of the names in the local name table if an * is in the first byte of the NCB_NAME field. The minimum buffer length possible is 4 bytes.

Note: A return code of hex 06 is returned in the NCB_RETCODE field if the receive buffer is not large enough for the data being sent. The remaining data is lost at this point.

Fields Required:

NCB_BUFFER@
NCB_LENGTH
NCB_NAME (Specify an * for all names.)
NCB_POST@ (If no-wait option used.)
NCB_LANA_NUM (Number of the adapter for session status.)

Fields Returned:

NCB_RETCODE
NCB_LENGTH
NCB_CMD_CPLT

Immediate Return Codes:

Hex 00 —Command has no immediate errors
Hex 03 —Invalid command
Hex 15 —Invalid name
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

Final Return Codes:

Hex 00 —Command completed
Hex 01 —Invalid buffer length
Hex 03 —Invalid command
Hex 06 —Message incomplete
Hex 19 —Name conflict detected
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

Data areas returned contain the following:

- Name number of sessions being reported on—1 byte
- Number of sessions with this name—1 byte
- Number of Receive Datagram and Receive Broadcast Datagram commands outstanding—1 byte
- Number of Receive Any commands outstanding—1 byte
- Information returned about a session—36 bytes per session
  - Local session number—1 byte
  - State of the session—1 byte

  This byte is represented as follows:

LISTEN pending	hex 01
CALL pending	hex 02
Session established	hex 03
HANG UP pending	hex 04
HANG UP complete	hex 05
Session Ended Abnormally	hex 06

  - Local name—16 bytes
  - Remote name—16 bytes
  - Number of Receive commands outstanding—1 byte
  - Number of Send and Chain Send commands outstanding—1 byte

Datagram Support

Datagram support commands allow you to send a message to a name or a group name, or to broadcast a message to everyone on the network. These commands also allow you to receive a datagram message from a name or a group name, or from anyone on the network. Datagram support differs from session support in several ways. The message is never acknowledged by the receiver, so it is up to the sender and receiver to agree on their own network protocols. Message lengths range up to 512 bytes with the Send command. If you specify more than 512 bytes for a Receive Datagram or Receive Broadcast command you will receive only the maximum that is allowed for a Send Datagram or Send Broadcast command.

Datagrams are smaller than session Send commands and require additional protocol interaction for reliable data transmissions. For reliable transmissions, use session support.

SEND DATAGRAM

Command Code:

Hex 20 —Wait option
Hex A0 —No-wait option.

Function:

This command sends a datagram to a unique name or group name for receipt at a local node or remote node.

Fields Required:

NCB_BUFFER@
NCB_LENGTH
NCB_NUM
NCB_CALLNAME
NCB_POST@ (If no-wait option used)
NCB_LANA_NUM (Number of the adapter for the send datagram.)

Field Returned:

NCB_RETCODE
NCB_CMD_CPLT

Immediate Return Codes:

Hex 00 —Command has no immediate errors
Hex 03 —Invalid command
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

Final Return Codes:

Hex 00 —Command completed
Hex 03 —Invalid command
Hex 01 —Invalid buffer length
Hex 13 —Invalid name number
Hex 19 —Name conflict detected
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

SEND BROADCAST DATAGRAM

Command Code:

Hex 22 —Wait option
Hex A2 —No-wait option

Function:

This command sends a message to everyone who has a Receive Broadcast Datagram command outstanding. If the remote adapter does not have a Receive Broadcast Datagram command outstanding, it does not get the message. If a computer issues a Send Broadcast Datagram command and has a Receive Broadcast Datagram command outstanding, the adapter receives its own message. If the adapter has several broadcast messages pending, the next Send Broadcast command issued satisfies all Receive Broadcast commands.

Fields Required:

NCB_BUFFER@
NCB_LENGTH
NCB_NUM
NCB_POST@ (If no-wait option used)
NCB_LANA_NUM (Number of the adapter for the send broadcast datagram.)

Field Returned:

NCB_RETCODE
NCB_CMD_CPLT

Immediate Return Codes:

Hex 00 —Command has no immediate errors
Hex 03 —Invalid command
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

Final Return Codes:

Hex 00 —Command completed
Hex 01 —Invalid buffer length
Hex 03 —Invalid command
Hex 13 —Invalid name number
Hex 19 —Name conflict detected
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

RECEIVE DATAGRAM

Command Code:

Hex 21 —Wait Option (Use with care since all processing halts until the datagram is received.)
Hex A1 —No-wait option

Function:

This command receives a datagram message from any name on the network directed to you. There is no time-out associated with this command. If you do not have a Receive Datagram command outstanding at the time a Send Datagram command is issued, you will not receive the data.

This command does not receive a broadcast datagram but will receive a datagram sent to a group name.

Note: A return code of hex 06 is returned in the NCB_RETCODE field if the receive buffer is not large enough for the data being sent. The remaining data is lost at this point.

Fields Required:

NCB_BUFFER@
NCB_LENGTH
NCB_NUM (If this field is hex FF, then receive a datagram from any other name for any of your names.)
NCB_POST@ (If no-wait option used.)
NCB_LANA_NUM (Number of the adapter for the receive datagram.)

Fields Returned:

NCB_RETCODE
NCB_LENGTH
NCB_CALLNAME
NCB_CMD_CPLT

Immediate Return Codes:

Hex 00 —Command has no immediate errors
Hex 03 —Invalid command
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

Final Return Codes:

Hex 00 —Command completed
Hex 03 —Invalid command
Hex 06 —Message incomplete
Hex 0B —Command canceled
Hex 13 —Invalid name number
Hex 17 —Name deleted
Hex 19 —Name conflict detected
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

RECEIVE BROADCAST DATAGRAM

Command Code:

Hex 23 —Wait option (Use with care since all processing halts until the datagram is received.)
Hex A3 —No-wait option

Function:

This command receives a message from anyone who issues a Send Broadcast Datagram command. There is no time-out for this command.

Note: A return code of hex 06 is returned in the NCB_RETCODE field if the receive buffer is not large enough for the data being sent. The remainder of the data is lost.

Fields Required:

NCB_BUFFER@
NCB_LENGTH
NCB_NUM
NCB_POST@ (If the no-wait is option used.)
NCB_LANA_NUM (Number of the adapter for the receive broadcast datagram.)

Fields Returned:

NCB_RETCODE
NCB_LENGTH
NCB_CALLNAME
NCB_CMD_CPLT

Immediate Return Codes:

Hex 00 —Command has no immediate errors
Hex 03 —Invalid command
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

Final Return Codes:

Hex 00 —Command completed
Hex 03 —Invalid command
Hex 06 —Message incomplete
Hex 0B —Command canceled
Hex 13 —Invalid name number
Hex 17 —Name deleted
Hex 19 —Name conflict detected
Hex 21 —Interface busy
Hex 22 —Too many commands outstanding
Hex 23 —Invalid number in NCB_LANA_NUM field
Hex (40 – 4F) —Unusual network condition
Hex (50 – FE) —Adapter malfunction

NCB Command Overview

The following summarizes the NCB commands. For commands not described below or for any additional information on a command, refer to the specific implementation of the NETBIOS that you are using.

Alphabetic List of NCB Commands

Command	Command Code (Hex)	
	No-Wait	**Wait**
Adapter Status	B3	33
Add Name	B0	30
Add Group Name	B6	36
Cancel		35
Call	90	10
Chain Send	97	17
Delete Name	B1	31
Hang Up	92	12
Listen	91	11
Receive	95	15
Receive Any	96	16
Receive Broadcast Datagram	A3	23
Receive Datagram	A1	21
Reset		32
Send	94	14
Send Broadcast Datagram	A2	22
Send Datagram	A0	22
Session Status	B4	34
Unlink		70

Figure 1-4. Alphabetic List of NCB Commands

Numeric List of NCB Commands

Command Code (Hex)	Command	State
10	Call	Wait
11	Listen	Wait
12	Hang Up	Wait
14	Send	Wait
15	Receive	Wait
16	Receive Any	Wait
17	Chain Send	Wait
20	Send Datagram	Wait
21	Receive Datagram	Wait
22	Send Broadcast Datagram	Wait
23	Receive Broadcast Datagram	Wait
30	Add Name	Wait
31	Delete Name	Wait
32	Reset	Wait
33	Adapter Status	Wait
34	Session Status	Wait
35	Cancel	Wait
36	Add Group Name	Wait
70	Unlink	Wait
90	Call	No-wait
91	Listen	No-wait
92	Hang Up	No-wait
94	Send	No-wait
95	Receive	No-wait
96	Receive Any	No-wait
97	Chain Send	No-wait
A0	Send Datagram	No-wait
A1	Receive Datagram	No-wait
A2	Send Broadcast Datagram	No-wait
A3	Receive Broadcast Datagram	No-wait
B0	Add Name	No-wait
B1	Delete Name	No-wait
B3	Adapter Status	No-wait

Figure 1-5. Numeric List of NCB Commands

NCB Command Summary

The following figure summarizes the NCB commands.

Command	Command Code (Hex)		NCB Fields	
	No-Wait	Wait	Required	Returned
Reset		32	1,2,4,12	2,13
Cancel		35	1,5,12	2,13
Adapter Status	B3	33	1,5,6,7,(11),12	2,6,13
Unlink		70	1,12	2,13
Add Name	B0	30	1,8,(11),12	2,4,13
Add Group Name	B6	36	1,8,(11),12	2,4,13
Delete Name	B1	31	1,8,(11),12	2,13
Call	90	10	1,7,8,9,10,(11),12	2,3,13
Listen	91	11	1,7,8,9,10,(11),12	2,3,(7),13
Hang Up	92	12	1,3,(11),12	2,13
Send	94	14	1,3,5,6,(11),12	2,13
Chain Send	97	17	1,3,5,6,7,(11),12	2,13
Receive	95	15	1,3,5,6,(11),12	2,6,13
Receive Any	96	16	1,3,4,5,6,(11),12	2,3,(4),6,13
Session Status	B4	34	1,5,6,8,(11),12	2,6,13
Send Datagram	A0	20	1,4,5,6,7,(11)	2,13
Send Broadcast Datagram	A2	22	1,4,5,6,(11),12	2,13
Receive Datagram	A1	21	1,4,5,6,(11),12	2,6,7,13
Receive Broadcast Datagram	A3	23	1,5,6,(11),12	2,6,7,13

Note: See previous text for more information if parentheses appear around a required or returned field.

NCB Fields

1. NCB_COMMAND
2. NCB_RETCODE
3. NCB_LSN
4. NCB_NUM
5. NCB_BUFFER@
6. NCB_LENGTH
7. NCB_CALLNAME
8. NCB_NAME
9. NCB_RTO
10. NCB_STO
11. NCB_POST@
12. NCB_LANA_NUM
13. NCB_CMD_CPLT
14. NCB_RESERVE

Figure 1-6. NCB Command Summary

Return Code Summary

The following figures summarize the NCB return codes. For return codes not described below, refer to the specific implementation of the NETBIOS that you are using.

Numeric List of Return Codes

The following figure lists the NETBIOS return codes in numeric order.

Hex Value	Return Code Name
00	Command complete or command has no immediate errors
01	Invalid buffer length
03	Invalid command
05	Command timed out
06	Message incomplete
08	Invalid local session number
09	No resource available
0A	Session closed
0B	Command canceled
0D	Duplicate name in local name table
0E	Name table full
0F	Command completed (name has active sessions and is now deregistered)
11	Local session table full
12	Session open rejected
13	Invalid name number
14	No answer (cannot find name called)
15	Invalid name
16	Name in use on remote node
17	Name deleted
18	Session ended abnormally
19	Name conflict detected
21	Interface busy
22	Too many commands outstanding
23	Invalid number in NCB_LANA_NUM field
24	Command completes while cancel is occurring
26	Command not valid to cancel
40	Adapter malfunction
50 to F6	Adapter malfunction
FF	Command pending status

Figure 1-7. List of NCB Return Codes

NCB Return Code Causes and Actions

The following lists the return-code causes and recommended actions.

00 Command completed or Command has no immediate errors

Cause: Command completed or command has no immediate errors

Action: No action required. This is normal after each successful command.

01 Invalid buffer length for SEND DATAGRAM, SEND BROADCAST DATAGRAM, ADAPTER STATUS, or SESSION STATUS.

Cause: A Send Broadcast or Send Datagram command cannot send more than 512 bytes. For ADAPTER STATUS and SESSION STATUS, the buffer length specified was less than the minimum required.

Action: Specify the correct size for the buffer and try again.

03 Invalid command

Cause: The command code used was incorrect.

Action: Reissue the correct command code.

05 Command timed out

Cause:

1. ADAPTER STATUS: The system time-out period has elapsed.
2. SEND: The time-out period specified for the Call or Listen command has elapsed.
3. RECEIVE: The time-out period specified for the Call or Listen command has elapsed.
4. HANG UP: The time-out period has expired for any outstanding Send commands to complete.

Action:

1. ADAPTER STATUS: Make sure you are using the correct name.

2. SEND: The session ended abnormally. Establish another session and reissue a Send command.
3. RECEIVE: Reissue the command.
4. HANG UP: The session ended abnormally.

06 Message incomplete

Cause: Part of a message was not received because the specified buffer length is not large enough to receive the full message.

Action: Reissue another Receive or Receive Any command to get the rest of the message before the remote node times out under session support. For Adapter Status, Session Status, Receive Datagram, and Receive Broadcast Datagram commands, the remaining data is lost.

08 Invalid local session number

Cause: The session number specified is not one of the active sessions.

Action: Specify an active session and reissue the command.

09 No resource available

Cause: Not enough space available on the node

Action: Reissue the command at a later time. This is a temporary condition.

0A Session closed

Cause: The session has been closed by either the local or remote node.

Action: No action is required. For a pending SEND, RECEIVE, or RECEIVE ANY, this is the notification that the session has been closed. For HANG UP, the session was closed by the remote computer.

0B Command canceled

Cause: Notification that the command was canceled. If the canceled command is SEND or CHAIN SEND the session is abnormally ended.

Action: No action is required.

0D Duplicate name in local name table

Cause: A name was specified that is already in the local name table.

Action: Specify another name.

0E Name table full

Cause: The maximum number of names is in the name table. The maximum number of names depends on the specific NETBIOS implementation you are using.

Action: Wait until a delete name occurs or deregistration is complete.

0F Command completed (name has active sessions and is now deregistered)

Cause: The name to be deleted is active in a session now, but is deregistered. When the name is marked deregistered and has active sessions, it still occupies a slot in the name table. The name is unusable.

Action: Close all sessions using that name so the Delete command can complete.

11 Local session table full

Cause: No entries are available in the session table. (The number of sessions for a local session table is user-specifiable and has maximum values depending on the specific NETBIOS implementation you are using.)

Action:

- Wait until a session has closed.
- Refer to the Reset command to alter values.

12 Session open rejected

Cause: No Listen command is outstanding on the remote node.

Action: Wait until a Listen command is issued by the remote node.

13 Invalid name number

Cause: The name number is invalid.

Action: Use the original name number that was assigned to the name.

14 No answer (cannot find name called)

Cause: The call name specified cannot be found or did not answer.

Action:

1. Verify that the call name used is correct.
2. Retry with the correct or a different call name.
3. Reissue if the remote node is busy.

15 Name not found

Cause: One of the following has occurred:

- The name specified was not in the name table.
- An asterisk was specified in the first character position of the name field.
- Hex 00 was specified in the first character position.
- The name is deregistered.

Action: Specify another name.

16 Name in use on remote node

Cause: Unique names may be used only once on the network.

Action: Specify another name.

17 Name deleted

Cause: The name that issued a Receive Datagram, Receive Broadcast Datagram, Listen, or Receive Any command has been deleted.

Action: No action required.

18 Session ended abnormally

Cause: One of the following has occurred:

- The remote node is powered off.
- The session SEND or CHAIN SEND has timed out.
- SEND or CHAIN SEND was canceled.
- HANG UP timed out while waiting for SEND or CHAIN SEND to complete.

Action:

- Check the remote node for status.
- Reestablish the session for a Send, Chain Send, Receive, or Receive Any command.

19 Name conflict detected

Cause: Network protocol has detected two or more identical names on the network.

Action: Every node on the network should delete that name immediately.

21 Interface busy

Cause: BIOS was called while executing an uninterruptible process.

Action: Return from the interrupt level and try again later.

22 Too many commands outstanding

Cause: The number of outstanding commands is maximum.

Action:

- If not at the maximum number, refer to RESET.
- If at the maximum number, try again later.

23 Invalid number in NCB _ LANA _ NUM field

Cause: A number other than hex 00 or hex 01 was specified.

Action: Correct the number and try again. Specify hex 00 for the first adapter or hex 01 for the second adapter.

24 Command completes while cancel is occurring

Cause: An attempt was made to cancel a command that already completed.

Action: No action required.

26 Command not valid to cancel

Cause: An attempt was made to cancel a command that is invalid to cancel.

Action: The commands that are not valid to cancel are: ADD NAME, ADD GROUP NAME, DELETE NAME, SEND DATAGRAM, SEND BROADCAST DATAGRAM, SESSION STATUS, RESET, and CANCEL.

40 Adapter malfunction

Cause: A hardware problem was detected on the adapter

Action: Retry or reset the command. If you receive the return code again, perform the node diagnostic tests.

50 to F6 Adapter malfunction

Cause: A hardware problem was detected on the adapter.

Action: Retry or reset the command. If you receive the return code again, perform the node diagnostic tests.

FF Command pending status

Cause: The command is still pending.

Action: No action is required. See NCB_POST@ and NCB_RETCODE for a description of this return code.

Section 2. IBM PC Network Adapter

The IBM PC Network Adapter contains the NETBIOS in ROM. This provides the basis for all program control of the network. Data transfer is accomplished through a layered data transfer protocol structure. The four major groups of NETBIOS support functions (session support, datagram support, name support, and network adapter status and control) are accessible through interrupt hex 5C.

The NETBIOS supports up to 32 concurrent duplex sessions, 32 NCB commands, and 16 names on each adapter.

NCB Return Codes

The PC Network Adapter generates the following additional NETBIOS return codes.

1A **Incompatible remote device**

Cause: Unexpected protocol frame received.

Action: Verify that all nodes on the network agree with the network protocols.

41 **Hot carrier detected**

Cause: The adapter detected a hot carrier on the network.

Action: Perform the node diagnostic tests.

42 **Hot carrier sent**

Cause: The adapter sent a hot carrier on the network.

Action: Perform the node diagnostic tests.

43 **No carrier detected**

Cause: No carrier was detected by the adapter.

Action: Perform the node diagnostic tests.

44 to 4D Unusual network condition

Cause: The NETBIOS detected an unusual condition on the network.

Action: Retry or reset the command. If you receive the return code again, perform the node diagnostic tests.

The following summarizes the commands that use the additional PC Network Adapter return codes.

Return Code	NCB Command	Type
1A	CALL	Final Return Code
	LISTEN	Final Return Code

Figure 2-1. PC Network Adapter NCB Return Codes

Status (Adapter Status) Command

Data that is returned in the data buffer field contains the following information.

Unit identification number—6 bytes: This number is part of the permanent node name. The unit identification number is represented as follows:

Byte 0: Low word, low byte
Byte 1: Low word, high byte
Byte 2: Middle word, low byte
Byte 3: Middle word, high byte
Byte 4: High word, low byte
Byte 5: High word, high byte.

External option status—1 byte: The status of the PC Network Adapter is represented below:

Bit	Function
7	Jumper W2 1 = Jumper W2 on 0 = Jumper W2 off
6	Jumper W1 1 = Jumper W1 on 0 = Jumper W1 off
5	Reserved
4	Reserved
3	Reserved
2	Reserved
1	Reserved
0	Reserved

Figure 2-2. PC Network Adapter Option Byte

Results of last self-test—1 byte: The result of the PC Network Adapter self-test is shown. The possible results are:

Result	Meaning
80	Successful Completion
81	Processor Test Failed
82	ROM Checksum Test Failed

Figure 2-3 (Part 1 of 2). Results of Last PC Network Adapter Self-Test

Result	Meaning
83	Unit ID PROM Test Failed
84	RAM Test Failed
85	Host Interface Test Failed
86	+/- 12V Test Failed
87	Digital Loopback Test Failed
8D	82588 Time-out
8E	Possible Constant Carrier
8F	Analog Loopback Test Failed

Figure 2-3 (Part 2 of 2). Results of Last PC Network Adapter Self-Test

Software version—2 bytes: The software version of the protocol layers are represented as follows:

Byte 0: Major version number
Byte 1: Minor version number.

Traffic and error statistics—48 bytes:

1. Duration of reporting period (in minutes)—2 bytes. After the counter reaches a value of hex 0FFFF, it resets to 0.
2. The number of CRC errors received—2 bytes. After the counters reach a value of hex 0FFFF, they do not increment further.[1]
3. The number of alignment errors received—2 bytes. After the counters reach a value of hex 0FFFF, they do not increment further.[1]
4. The number of collisions encountered—2 bytes. After the counter reaches a value of hex 0FFFF, it resets to 0.
5. The number of unsuccessful transmissions—2 bytes. After the counter reaches a value of hex 0FFFF, it resets to 0.[1]
6. Number of successfully transmitted frames—4 bytes. After the counter reaches a value of hex 0FFFFFFFF, it resets to 0.
7. Number of successfully received frames—4 bytes. After the counter reaches a value of hex 0FFFFFFFF, it resets to 0.
8. Number of retransmissions—2 bytes. After the counter reaches a value of hex 0FFFF, it resets to 0.

---

[1] See Intel's Microcommunications Handbook for further information.

9. Number of times the receiver exhausted its resources—2 bytes. After the counters reach a value of hex 0FFFF, they do not increment further.[2]

Adapter resource statistics:

1. Reserved for internal use—8 bytes.
2. Number of NCBs currently available for use—2 bytes.
3. Maximum number of NCBs supported by the adapter as configured by the Reset command—2 bytes.
4. Maximum number of NCBs possible for the adapter to support—2 bytes.
5. Reserved for internal use—4 bytes.
6. Pending sessions—2 bytes. A pending session is either a CALL-pending, a LISTEN-pending, a session established, a session ended abnormally, HANG UP-pending, or HANG UP-complete.
7. Maximum number of sessions supported by the adapter as configured by Reset command—2 bytes.
8. Maximum number of sessions possible for the adapter to support—2 bytes
9. Maximum session data frame size—2 bytes.

Quantity of names in the local name table—2 bytes: This number is the quantity of names present in the local name table.

Local name table—16 entries of 18 bytes each: The first 16 bytes of each entry represent the name, and the last 2 bytes represent the name status. This first byte is equal to the name number. The second byte denotes the status when it is masked with a hex 87. The mask is used to get the most-significant bit and the last 3 bits of the byte. The other bits are reserved and can have nonzero values.

- NXXXX000 = Trying to register a name
- NXXXX100 = A registered name
- NXXXX101 = A deregistered name

---

[2] See Intel's Microcommunications Handbook for further information.

- NXXXX110 = A detected duplicate name
- NXXXX111 = A detected duplicate name with deregister pending.

  Where:
  - X = Reserved bit
  - N = 0 The name is a unique name
  - N = 1 The name is a group name.

Programming Considerations

Two Network Adapters in the Same Computer

If installing two network adapters in the same computer the NCB_LANA_NUM field is hex 00 when issuing an NCB to the primary adapter, and hex 01 when issuing an NCB to the alternate adapter.

Adapter Presence Test

To check for a working PC Network adapter, check interrupt vector hex 5C.

- If the location contains all binary zeros, an adapter is not present.
  - If the location contains a value other than all binary zeros, issue either command hex 7F or hex FF.
  - If a code of hex 03 is returned in AL, the adapter is present.
- If the code returned in AL is one of the following, the adapter is present.
  - Hex 23 —Invalid number in NCB_LANA_NUM field
  - Hex (40 – 4F) —Unusual network condition
  - Hex (50 – FE) —Adapter malfunction

Multitasking

When using the wait option for the commands in a multitasking environment, a "hook" is provided for the multitasking program using interrupt hex 15. When either a busy or wait loop occurs in NETBIOS, a hook is provided for the program to break out of the loop. The register pair ES:BX is used to distinguish individual calls. This register points to the NCB. The hook is also used when NETBIOS is servicing an interrupt causing a corresponding wait loop, and thus providing a means of breaking out of the loop.

Programs in the multitasking environment have the responsibility to check the AX register for the following function codes:

AH contents:	AL contents:
90	80
91	80

Figure 2-4. AX Register Function Codes (Hex)

The program passes all other functions to the interrupt handler. This can be accomplished by either a JMP or a CALL command. With either a hex 90 or 91 in the AH register, the program performs the necessary processing and returns using an IRET instruction. A hex 80 in the AL register indicates that NETBIOS issued the interrupt.

Hex 9080 This function code is in the AX register whenever NETBIOS is about to enter either a busy or a wait loop. NETBIOS also issues an interrupt hex 15 at this time to signal the program of the loop. When this occurs, the program saves the task status and dispatches another task. This allows overlapping execution of tasks when the hardware is busy.

Hex 9180 This function code is in the AX register whenever NETBIOS has set an interrupt flag for a corresponding busy loop. The NETBIOS also issues an interrupt hex 15 at this time. This code is used to signal a POST condition, and the program sets the task status to "ready to run" before returning.

PC Network Adapter Pseudocode for NCB Commands

The following is the pseudocode for the NCB commands.

RESET

```
PROCEDURE Reset (command_block);
        {set configuration parameters}
BEGIN
IF either of the two configuration
    parameters is equal to zero THEN;
    set configuration parameter to zero
    {6 sessions, and 12 commands}
ELSE
IF either of the two configuration parameters is greater
    than maximum value
    set configuration parameter to maximum value
    BEGIN
    set configuration parameters in configuration table;
    return command_completed status to user;
    END
END;
```

STATUS

```
PROCEDURE Status (command_block);
    {get LANA configuration parameters and status}
BEGIN
IF buffer length is invalid THEN;
    return invalid_buffer_length status to user;
ELSE
    IF name in conflict THEN;
       return name_conflict status to user;
    ELSE
        IF name is deregistered THEN;
           return invalid_name to user;
    ELSE
        IF name is a local name or name is * THEN;
        BEGIN
        get configuration parameters and status of local LANA;
        return configuration parameters and status;
        return actual length of configuration
            parameter and status;
        IF size of user buffer is smaller than the configuration
            parameter and status THEN;
            return message_incomplete status to user;
        ELSE
            return command_completed status to user;
        END
    ELSE
        BEGIN
        send status_request frame to remote node;
        wait for response from remote node;
            {remote node send status_response frame}
        IF status_response frame is received within the time-out
            interval THEN;
            BEGIN
            extract configuration parameters and
                status of remote LANA;
            return configuration parameters and status;
            return actual length of configuration
                parameter and status;
            IF size of user buffer is smaller than the configuration
                parameter and status THEN;
                return message_incomplete status to user;
            ELSE
                return command_completed status to user;
            END
        ELSE
            return command_timed_out status to user;
        END
END;
```

ADD NAME

```
PROCEDURE Add_Name (command_block);
        {request local registration of name}
BEGIN
IF name begins with * or null THEN;
    return invalid_name status to user;
ELSE
    BEGIN
    search for name in local name table;
    IF name is found THEN;
        BEGIN
            IF name in conflict THEN;
               return name_conflict status to user;
            ELSE
              IF name is deregistered THEN;
                 return invalid_name status to user
            ELSE return existing name_number to user;
                 return duplicate_name status to user;
        END
    ELSE
        IF local name table is full THEN;
            return name_table_full status to user;
        ELSE
            BEGIN
              IF no resource for transmission THEN;
                 return no_resource_available status to user
            REPEAT
            ELSE
                BEGIN
                broadcast name_claim frame to network;
                wait for response from remote node in network;
                {all remote nodes search for name
                in their local name tables and send
                name_claim_response frame if found}
                END
            UNTIL name_claim_response frame is received
                or number of times to broadcast is reached;
            IF name_claim_response frame is not received THEN;
                BEGIN
                enter name to local name table;
                return name_number to user;
                return command_completed status to user;
                END
            ELSE
                return name_in_use status to user;
            END
    END
END;
```

ADD GROUP NAME

```
PROCEDURE Add_Group_Name (command_block);
       {request local registration of name}
BEGIN
IF name begins with * or null THEN;
    return invalid_name status to user;
ELSE
    BEGIN
    search for name in local name table;
    IF name is found THEN;
        BEGIN
            IF name in conflict THEN;
                return name_conflict status to user;
            ELSE
               IF name is deregistered THEN;
                 return invalid_name status to user;
            ELSE
            return existing name_number to user;
            return duplicate_name status to user;
        END
    ELSE
        IF local name table is full THEN;
            return name_table_full status to user;
        ELSE
            BEGIN
            If no resource available for transmission THEN;
               return no_resource_available status to user
            ELSE
            REPEAT
                BEGIN
                broadcast add_group_name_claim frame to network;
                wait for response from remote node in network;
                {all remote nodes search for name in their local name
                tables and send name_claim_response
                frame if found}
                END
            UNTIL name_claim_response frame is received or
                number of times to broadcast is reached;
            IF name_claim_response frame is not
                received THEN;
                BEGIN
                enter name to local name table;
                return name_number to user;
                return command_completed status to user;
                END
            ELSE
                return name_in_use status to user;
            END
    END
END;
```

DELETE NAME

```
PROCEDURE Delete_Name (command_block);
        {request local deregistration of name}
BEGIN
IF name begins with * or null THEN;
    return invalid_name status to user;
ELSE
    BEGIN
    search for name in local name table;
    IF name is found THEN;
        BEGIN
        check for pending nonactive session commands;
        IF nonactive session command is found THEN;
            terminate the nonactive session command;
        check the session count in table entry;
        IF session count is zero THEN;
            BEGIN
            remove name from local name table;
            return command_complete status to user;
            END
        ELSE
            BEGIN
            change status of name to deregistered;
                {name will be removed from local name
                table when session count reaches zero}
            return command_completed_name_has_
                active_session status to user;
            END
        END
    ELSE
        return invalid_name status to user;
    END
END;
```

CALL

```
PROCEDURE Call (command_block);
    {open user session with remote name using name supplied}
BEGIN
IF local resource is not available THEN;
    return no_resource_available status to user;
ELSE
    BEGIN
    search for name in local name table;
    IF name is not found or name is deregistered THEN;
        return invalid_name status to user;
    ELSE
    IF name is marked "conflict detected" THEN;
        return name_conflict_detected status to user;
    ELSE
        IF local session table is full THEN;
            return local_session_table_full status to user;
        ELSE
            BEGIN
            search for remote name in local name table;
                BEGIN
                REPEAT
                    BEGIN
                    IF name is not found THEN
                        broadcast name_query frame to network;
                    ELSE loopback name_query frame;
                        wait for response from remote node or
                        time-out;
                        {all remote nodes search for name
                        in their local name tables and send
                        name_query_response frame if found}
                    END
                UNTIL name_query_response frame is received
                    or number of times to broadcast is
                    reached;
                IF name_query_response frame is not
                    received THEN;
                    return unknown_remote_name status to user;
                END
                BEGIN
                send session_request frame to destination
                    node;
                    {destination node search for pending
                    LISTEN command, if found, return
                    session_accept frame, else, return
                    session_reject frame}
                IF network error THEN;
                    return session_aborted to user;
                IF response from destination is received
                    within the time-out interval THEN;
                    CASE response OF
```

```
                    session_accept frame:
                        BEGIN
                        set session_established indicator
                            in session table;
                        return local_session_number
                            to user;
                        return command_completed status
                            to user;
                        END
                        {session established}
                    session_reject frame:
                        return session_open_rejected status
                            to user;
                END {case}
            ELSE
                return command_timed_out status to user;
            END
        END
    END
END;
```

LISTEN

```
PROCEDURE Listen (command_block);
    {open user session with remote name, using
    name supplied}
BEGIN
IF local resource is not available THEN;
    return no_resource_available status to user;
ELSE
    BEGIN
    search for name in local name table;
    IF name is not found or name is deregistered THEN;
        return invalid_name status to user;
    ELSE
        IF name is marked "conflict detected" THEN
            return name_conflict_detected status to user;
        ELSE
            IF local session table is full THEN;
                return local_session_table_full status to user;
            ELSE
                BEGIN
                REPEAT
                    BEGIN
                    wait for session_request frame
                        or name_conflict_detection;
                    IF LISTEN specific is specified THEN;
                        check source of session_request frame;
                        IF source of session_request frame is
                            same as remote name THEN;
                            set session_request_completed indicator;
                            {LISTEN specific satisfied}
                        ELSE
                            reset session_request_completed
                                indicator;
                            {LISTEN not satisfied, continue
                            to wait}
                    ELSE
                        IF LISTEN any specified THEN
                           set session_request_completed indicator;
                           {LISTEN ANY satisfied}
                    END

                UNTIL session_request_completed indicator is set
                    or name_conflict_detected;
                IF name_conflict_detected for outstanding
                    LISTEN THEN
                ELSE IF network error THEN;
                    return session_aborted status to user;
                     ELSE
                         send session_accept frame to source;
                         wait for first frame on session;
                         set session_established
```

```
                    indicator in session table;
                return source of session_request
                    frame to user;
                return local_session_number to user;
                return command_completed status to user;
                {session established}
            END
        END
END;
```

HANG UP

```
PROCEDURE Hang Up (command_block);
    {close user session indicated by
    local_session_number}
BEGIN
IF local session number is invalid THEN;
    return invalid_session_number status to user;
ELSE
    IF session is already closed and session_closed status has
        not been reported THEN;
        return session_closed status to user;
    ELSE
        IF session is already aborted and session_aborted
            status has not been reported THEN;
            return session_aborted status to user;
        ELSE
            BEGIN
            REPEAT
                IF a RECEIVE command is pending THEN;
                    terminate RECEIVE command with a
                    session_closed status to user;
            UNTIL all pending RECEIVE commands
                are terminated;
            REPEAT
                IF a SEND or CHAIN SEND command
                    is pending THEN;
                    wait until the SEND, CHAIN SEND, or
                        HANG UP has completed or
                        timed out;
                IF the SEND command was timed out THEN
                    abort the session;
            UNTIL all pending SEND and CHAIN SEND
                commands are completed or timed out;
            send close frame to destination node;
            wait for close frame from destination node or
                close time-out;
                {destination node close the session
                and send close frame}
            IF close frame is received before the close
                time-out interval THEN;
                BEGIN
                close the session;
                return command_completed status to user;
                END
            ELSE
                BEGIN
                abort the session;
                return session aborted status to user;
                END
            IF RECEIVE ANY to name command is pending THEN;
                terminate RECEIVE ANY to name command with
```

```
            session_closed or session_aborted status;
        ELSE
            IF a RECEIVE ANY to any name command
               is pending THEN
                terminate the RECEIVE ANY to any name command
                with session_closed or
                session_aborted status;
        END
END;
```

SEND

```
PROCEDURE Send (command_block);
    {send data through user session as indicated by
    local_session_number}
BEGIN
IF local session number is invalid THEN;
    return invalid_local_session_number
    status to user;
ELSE
    IF session is closed and session_closed status has not
        been reported THEN;
        return session_closed status to user;
    ELSE
        IF session is aborted and session_aborted status has
            not been reported THEN;
            return session_aborted status to user;
        ELSE
            BEGIN
            send session data frame(s) to destination node;
            wait for ack frame(s) from destination node
                or for time-out;
            IF session data is sent successfully within the
                time-out interval for session send THEN;
                return command_completed status to user;
            ELSE
                BEGIN
                abort the session;
                return command_timed_out status to user;
                END
            END
END;
```

CHAIN SEND

```
PROCEDURE Chain Send (command_block);
    {send data through user session as indicated by
    local_session_number}
BEGIN
IF local session number is invalid THEN;
    return invalid_local_session_number status to user;
ELSE
    IF session is closed and session_closed status
        has not been reported THEN;
        return session_closed status to user;
    ELSE
        IF session is aborted and session_aborted
            status has not been reported THEN;
            return session_aborted status to user;
        ELSE
            BEGIN
            send session data frame(s) to destination node;
            wait for ack frame(s) from destination node
                or for time-out;
            IF session data is sent successfully within the
                time-out interval for session send THEN;
                return command_completed status to user;
            ELSE
                BEGIN
                abort the session;
                return command_timed_out status to user;
                END
            END
END;
```

RECEIVE

```
PROCEDURE Receive (command_block);
    {receive data through user session as indicated by
    local_session_number}
BEGIN
IF local session number is invalid THEN;
    return invalid_local_session_number
        status to user;
ELSE
    IF session is closed and session_closed status has not
        been reported THEN;
        return session_closed status to user;
    ELSE
        IF session is aborted and session_aborted status has
            not been reported THEN;
            return session_aborted status to user;
        ELSE
            BEGIN
            wait for session message (data frame(s))
                from source node;
            IF session data is received within the time-out
                interval for session receive THEN;
                BEGIN
                send ack frames(s) to source node;
                transfer session data to user buffer
                    of appropriate length;
                return actual length of transfer to user;
                IF size of user buffer is smaller than
                    received session data THEN;
                    return message_incomplete status to user;
                ELSE
                    return command_completed status to user;
                END
            ELSE
                return command_timed_out status to user;
                {session data received}
            END
END;
```

RECEIVE ANY

```
PROCEDURE Receive_Any (command_block);
    {receive any data sent to the
    specified name_number}
BEGIN
IF name number is invalid or deregistered THEN;
    return invalid_name_number
        status to user;
ELSE
    IF session is closed and session_closed
        status has not been reported THEN;
        return session_closed status to user;
    ELSE
        IF session is aborted and session_aborted
            status has not been reported THEN;
            return session_aborted status to user;
        ELSE
            BEGIN
            REPEAT
                BEGIN
                wait for a session message (data frame(s));
                IF RECEIVE specific is specified THEN;
                    BEGIN
                    check recipient name in session message;
                    IF recipient name in session message is
                        same as local name THEN;
                        set receive_completed indicator;
                        {receive to specific name satisfied}
                    ELSE
                        reset receive_completed indicator;
                        {receive to specific not satisfied,
                        continue to wait}
                    END
                ELSE
                    set receive_completed indicator;
                    {receive to any name satisfied}
                END
            UNTIL receive_completed indicator is set;
            IF "conflict detected" error THEN;
                return name_conflict_detected status to user;
            ELSE
                send ack frame(s) to sending node;
                transfer session data to user buffer
                    of appropriate length;
                return actual length of transfer to user;
                return recipient name_number to user;
                return local_session_number to user;
                IF size of user buffer is smaller than
                    received session data THEN;
```

```
                return message_incomplete status to user;
            ELSE
                return command_completed status to user;
                {session data received}
        END
END;
```

SESSION STATUS

```
PROCEDURE Session_Status (command_block);
    {obtain status of session indicated by name}
BEGIN
IF buffer length is invalid THEN;
    return invalid_buffer_length status to user;
ELSE
    BEGIN
        IF name in conflict THEN
            return name_conflict_detected status to user;
        IF name does not start with * THEN;
            IF name is registered search for name in local name table;
            THEN return number of pending sessions;
            return number of pending datagram receives;
            IF name or * is found THEN;
                BEGIN
                get session status in session table
                    for each session on the name or;
                    for all session if * ;
                    return session status to user;
                    return actual length of session status;
                IF size of user buffer is smaller than
                    size of session status data THEN;
                    return message_incomplete status to user;
                ELSE
                    return command_completed status to user;
                END
        ELSE
            return invalid_name status to user;
        END
END;
```

SEND DATAGRAM

```
PROCEDURE Send Datagram (command_block);
    {send datagram to remote node with the specified
    name registration}
BEGIN
IF buffer length is invalid THEN;
    return invalid_buffer_length status to user;
ELSE
    BEGIN
    search for name corresponding to name number
        in local name table;
    IF name in conflict THEN;
        return name_conflict status to user;
    ELSE
        IF registered name is not found THEN;
            return invalid_name_number status to user;
        ELSE
            BEGIN
            send datagram to destination node;
            return command_completed status to user;
            END
    END
END;
```

SEND BROADCAST DATAGRAM

```
PROCEDURE Send_Broadcast_Datagram (command_block);
    {send broadcast datagram to all nodes on the network}
BEGIN
IF buffer length is invalid THEN;
    return invalid_buffer_length status to user;
ELSE
    BEGIN
    search for name in local name table;
    IF name in conflict THEN;
        return name_conflict status to user;
    ELSE
        IF registered name is not found THEN;
            return invalid_name_number status to user;
        ELSE
            BEGIN
            broadcast datagram to all nodes on the network;
            return command_completed status to user;
            END
    END
END;
```

RECEIVE DATAGRAM

```
PROCEDURE Receive Datagram (command_block);
    {receive datagram from any node on the network}
BEGIN
search for name corresponding to name number
    in local name table;
IF registered name is not found THEN;
    return invalid_name_number status to user;
ELSE
    BEGIN
    REPEAT
        BEGIN
        wait for arrival of datagram;
        IF datagram receive specific is specified THEN;
            BEGIN
            check recipient name in datagram message;
            IF recipient name in datagram message is same
                as name specified by local name
                number THEN;
                set receive_completed indicator;
                {datagram receive to
                specific name satisfied}
            ELSE
                reset receive_completed indicator;
                {datagram receive not satisfied,
                continue to wait}
            END
        ELSE
            set receive_completed indicator;
            {datagram receive to any name satisfied}
        END
    UNTIL receive_completed indicator is set;
    IF "conflict detected" error THEN;
        return name_conflict_detected status to user;
    ELSE
        transfer datagram data to user buffer
            of appropriate length;
        return actual length of transfer to user;
        return local name_number to user;
        return sender's name to user;
        IF size of user buffer is smaller than
            received datagram THEN;
            return message_incomplete status to user;
            {data received, unable to
            transfer entire message}
        ELSE
            return command_completed status to user;
            {datagram received}
    END
END;
```

RECEIVE BROADCAST DATAGRAM

```
PROCEDURE Receive Broadcast Datagram (command_block);
    {receive broadcast datagram from
    any node in the network}
BEGIN
search for name corresponding to name number in
    local name table;
IF registered name is not found THEN;
    return invalid_name_number status to user;
ELSE
    wait for arrival of broadcast datagram;
    IF "conflict detected" error THEN;
        return name_conflict_detected status to user;
    ELSE
        transfer datagram to user buffer
            of appropriate length;
        return actual length of transfer to user;
        return sender's name to user;
        IF size of user buffer is smaller than received
            datagram THEN;
            return message_incomplete status to user;
            {broadcast datagram received, unable to return
            entire message}
        ELSE
            return command_completed status to user;
            {broadcast datagram received}
END;
```

PC Network Adapter Frame Description

The following describes the protocol frames and their functions.

Field Definitions

The description of the fields within the protocol frame types follows.

Field	Meaning
ACK	This is an 8-bit field that includes the sequence number + 1 modulo 256 of the last correctly received frame.
ALGNERRS	This is a 16-bit field specifying the number of frames received with alignment errors.
ALIASNAME	This is a 16-byte field containing the name.
ALIASNR	This is an 8-bit field specifying the number that is assigned to a given name.
ALIASTAT	The low-order 4 bits specify the status of a name. See page 2-7 for the numbers that are returned in this field.
CLREAS	This is an 8-bit reason code indicating the reason for a connection being closed. CLREAS = hex 00 indicates a normal close.
CMDRESP	This is a 1-bit flag indicating whether the frame contains a command or a response to a command.
COLLISIONS	This is a 16-bit field specifying the number of transmitted frames that experienced collisions.
CONNID	This is a 16-bit identifier used to determine which session a frame is assigned to.
CRC	This is a 32-bit field containing the cyclic redundancy check for the frame according to the Autodin-II 32-bit polynomial generator.
CRCERRS	This is a 16-bit field specifying the number of CRC errors being reported.

DADDR This 48-bit field identifies the destination Link level address of the frame. A value of all binary 1's indicates a broadcast. All other adapter addresses will have the 16 highest bits set to 0. A value whose least-significant bits are hex FF indicates a group address of the form f(name).

DATA DATA is a variable length field containing user data.

DID DID is a 16-bit field that is incremented with each retransmission of a name query, name claim, and a get status frame.

DLEN This 16-bit field contains the length of the data field, in bytes, of the Link level frame.

DNAME This is a 16-bit field that identifies the name of a destination (ASCII characters).

DNCID This is a 16-bit field used with CONNID to determine the session for each frame or to identify a datagram.

DNODEID This is a 48-bit field indicating the Link level address of an intended destination.

EFD This is an 8-bit end-frame delimiter flag (hex 7E).

EOM This is a 1-bit end-of-message indicator. The bit marks the end of a user's logical message.

JUMPSTAT1 This is a 1-bit field indicating the status of jumper W1. When set to 1, jumper W1 is in place.

JUMPSTAT2 This is a 1-bit field indicating the status of jumper W2. When set to 1, jumper W2 is in place.

NACKREAS This is an 8-bit field indicating the reason why a frame was not successfully delivered. The following is a list of the codes that can be contained in this field

- Hex 00 — Standard NACK frame
- Hex 10 — Bad sequence number
- Hex 12 — Rejected by a higher level
- Hex 18 — NACK response to an unexpected open request
- Hex 20 — Incompatible RSP version

- Hex 22 – Bad service ID
- Hex 24 – Invalid RSP control information
- Hex 26 – No remote resources
- Hex 32 – Acceptable NACK frame
- Hex 34 – Name claim rejected
- Hex 35 – Name query rejected

NODEIDMASK This is a 6-byte mask used in a logical AND operation to get the destination node name.

NODEIDMATCH This is the 6-byte match value to use to request a response.

NRALIAS This is the 16-bit field specifying the number of names to follow.

PKTSIZE This is a 16-bit field indicating the maximum frame size that an initiating session node is willing to accept.

PNCID This is a 16-bit field used with CONNID to determine the network connection with which a frame is associated at a previous node, or to identify a datagram.

PNODEID This is a 48-bit field indicating the Link level address of the previous node.

POLL This is a 1-bit field which, when set to hex 08, indicates that a return frame should be generated back to the sender. Hex 00 indicates no poll.

RECVMSGS This is a 32-bit field specifying the number of frames that have been successfully received for a given period of time.

REPORTPD This is a 16-bit field specifying the period of time, in minutes, for the statistical data that was gathered.

RVAL This is an 8-bit value indicating the reason a requested session was rejected. RVAL = 3 indicates no matching Listen command. RVAL = 4 indicates an incompatible version.

SADDR This is a 48-bit field identifying the source Link level address of the frame.

SCONNID This is a 16-bit identifier used to determine the session number for a frame.

SEQ SEQ is an 8-bit session frame sequence number. It is incremented with each *new* data transmission.

SHORTFRAMES This is a 16-bit field specifying the number of short frames being reported.

SNAME This is a 16-bit field containing the name of a source node (specified in ASCII characters).

SNCID This is a 16-bit field used with CONNID to determine the network connection with which a frame is associated at a source node, or to identify a datagram.

SNODEID This is 48-bit field indicating the Link level address of a source node.

STD STD is an 8-bit start delimiter flag (hex 7E).

TRANSID This is a 16-bit field indicating the transaction ID for status and status response frames.

WIN WIN is a 4-bit field that indicates the number of frames beyond those already acknowledged that the sender is willing to accept.

XMITABRTS This is a 16-bit field specifying the number of transmitted frames that were ended abnormally.

XMITMSGS This is a 32-bit field specifying the number of frames that were successfully transmitted for a given period of time.

XXXX The Xs represent a "don't-care" value.

Frame Types, Formats, and Functions

The following describes the protocol frame types and their functions.

Note: A "?" refers to a variable value that has meaning to the particular field for that frame.

Name Query Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 50XX	DW	Fixed value for this position of frame
N / A	11	Hex 10	DB	Fixed value for this position of frame
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept
CONNID	13	?	DW	Field ID for frame. Choose a new CONNID value on the first transmission or repeat the previous CONNID on retransmissions
N / A	15	Hex 0202	DW	Fixed value for this position of frame
N / A	17	Hex XXXX	DW	Don't-care value for position of frame
N / A	19	Hex 0100	DW	Fixed value for this position of frame
N / A	1B	Hex XXXX	DW	Don't-care value for position of frame

Figure 2-5 (Part 1 of 2). Name Query Frame

Field Name	Offset (Hex)	Length	Type	Description
N / A	1D	Hex XXXX	DW	Don't-care value for position of frame
N / A	1F	Hex 10XX	DW	Fixed value for this position of frame
N / A	21	Hex XX10	DW	Fixed value for this position of frame
DNAME	23	16 DUP(?)	DB	ASCII name for destination
SNAME	33	16 DUP(?)	DB	ASCII name for source
PNCID	43	?	DW	See definition
DID	45	?	DW	Number that is incremented by 1 for each frame
SNCID	47	?	DW	Field ID for frame
DNODEID	49	?	DW	Low address
	4B	?	DW	Mid address
	4D	?	DW	High address
SNODEID	4F	?	DW	Low address
	51	?	DW	Mid address
	53	?	DW	High address
PNODEID	55	?	DW	Low address
	57	?	DW	Mid address
	59	?	DW	High address
CRC	5B	?	DD	Check byte
EDF	5F	Hex 7E	DB	End-of-frame byte

Figure 2-5 (Part 2 of 2). Name Query Frame

Name Claim and Name Claim Cancel Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 50XX	DW	Fixed value for this position of frame
N / A	11	?	DB	Hex 10 = Name Claim frame Hex A0 = Name Claim cancel frame
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept
CONNID	13	?	DW	Field ID for frame. Choose a new CONNID value on the first transmission or repeat the previous CONNID on retransmissions.
N / A	15	Hex 0202	DW	Fixed value for this position of frame
N / A	17	Hex XXXX	DW	Don't-care value for this position of frame
N / A	19	Hex 0400	DW	Fixed value for this position of frame
N / A	1B	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1D	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1F	Hex 10XX	DW	Fixed value for this position of frame
N / A	21	Hex 0000	DW	Fixed value for this position of frame

Figure 2-6 (Part 1 of 2). Name Claim and Name Claim Cancel Frame

Field Name	Offset (Hex)	Length	Type	Description
DNAME	23	16 DUP(?)	DB	ASCII name for destination
PNCID	33	?	DW	Equal to SNCID
DID	35	?	DW	Number that is incremented by 1 for each frame
SNCID	37	?	DW	Field ID for frame
DNODEID	39	?	DW	Low address
	3B	?	DW	Mid address
	3D	?	DW	High address
SNODEID	3F	?	DW	Low address
	41	?	DW	Mid address
	43	?	DW	High address
PNODEID	45	?	DW	Low address
	47	?	DW	Mid address
	49	?	DW	High address
CRC	4B	?	DD	Check byte
EDF	4F	Hex 7E	DB	End-of-frame byte

Figure 2-6 (Part 2 of 2). Name Claim and Name Claim Cancel Frame

Nonexclusive Name Claim and Nonexclusive Name Claim Cancel Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01	?	DW	Low address
	03	?	DW	Mid address
	05	?	DW	High address
SADDR	07	?	DW	Low address
	09	?	DW	Mid address
	0B	?	DW	High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 50XX	DW	Fixed value for this position of frame
N / A	11	?	DB	Hex 10 = Nonexclusive Name Claim frame Hex A0 = Nonexclusive Name Claim Cancel frame
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept
CONNID	13	?	DW	Field ID for frame. Choose a new CONNID value on the first transmission or repeat the previous CONNID on retransmissions.
N / A	15	Hex 0202	DW	Fixed value for this position of frame
N / A	17	Hex XXXX	DW	Don't-care value for this position of frame
N / A	19	Hex 0600	DW	Fixed value for this position of frame
N / A	1B	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1D	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1F	Hex 10XX	DW	Fixed value for this position of frame

Figure 2-7 (Part 1 of 2). Nonexclusive Name Claim and Nonexclusive Name Claim Cancel Frame

Field Name	Offset (Hex)	Length	Type	Description
N / A	21	Hex 0000	DW	Fixed value for this position of frame
DNAME	23	16 DUP(?)	DB	ASCII name for destination
PNCID	33	?	DW	Equal to SNCID
DID	35	?	DW	Number that is incremented by 1 for each frame
SNCID	37	?	DW	Field ID for frame
DNODEID	39	?	DW	Low address
	3B	?	DW	Mid address
	3D	?	DW	High address
SNODEID	3F	?	DW	Low address
	41	?	DW	Mid address
	43	?	DW	High address
PNODEID	45	?	DW	Low address
	47	?	DW	Mid address
	49	?	DW	High address
CRC	4B	?	DD	Check byte
EDF	4F	Hex 7E	DB	End-of-frame byte

Figure 2-7 (Part 2 of 2). Nonexclusive Name Claim and Nonexclusive Name Claim Cancel Frame

Positive Name Query Response Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 30XX	DW	Fixed value for this position of frame
N / A	11	Hex 20	DB	Fixed value for this position of frame
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
SCONNID	15	?	DW	Connection ID of session
N / A	17	Hex XXXX	DW	Don't-care value for this position of frame
N / A	19	Hex 0101	DW	Fixed value for this position of frame
N / A	1B	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1D	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1F	Hex 10XX	DW	Fixed value for this position of frame
N / A	21	Hex 0010	DW	Fixed value for this position of frame
DNAME	23	16 DUP(?)	DB	ASCII name for destination
SNAME	33	16 DUP(?)	DB	ASCII name for source
DNCID	43	?	DW	Number used to determine the session for frame or to identify the datagram

Figure 2-8 (Part 1 of 2). Positive Name Query Response Frame

Field Name	Offset (Hex)	Length	Type	Description
SNCID	45	?	DW	Field ID for frame
SNODEID	47	?	DW	Low address
	49	?	DW	Mid address
	4B	?	DW	High address
CRC	4D	?	DD	Check byte
EDF	51	Hex 7E	DB	End-of-frame byte

Figure 2-8 (Part 2 of 2). Positive Name Query Response Frame

Negative Name Query Response Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01	?	DW	Low address
	03	?	DW	Mid address
	05	?	DW	High address
SADDR	07	?	DW	Low address
	09	?	DW	Mid address
	0B	?	DW	High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 40XX	DW	Fixed value for this position of frame
N / A	11	Hex 30	DB	Fixed value for this position of frame
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
N / A	15	Hex XXXX	DW	Don't care value for this position of frame
NACKREAS	17	?	DB	Reason why frame was not acknowledged
N / A	18	Hex XX	DB	Don't-care value for this position of frame
N / A	19	Hex 0101	DW	Fixed value for this position of frame
N / A	1B	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1D	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1F	Hex 10XX	DW	Fixed value for this position of frame
N / A	21	Hex XX10	DW	Fixed value for this position of frame
DNAME	23	16 DUP(?)	DB	ASCII name for destination

Figure 2-9 (Part 1 of 2). Negative Name Query Response Frame

Field Name	Offset (Hex)	Length	Type	Description
SNAME	33	16 DUP(?)	DB	ASCII name for source
DNCID	43	?	DW	Number to determine the session for the frame or to identify the datagram
CRC	45	?	DD	Check byte
EDF	47	Hex 7E	DB	End-of-frame byte

Figure 2-9 (Part 2 of 2). Negative Name Query Response Frame

Name Claim Response Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address This is the next node address (initiator's address)
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 40XX	DW	Fixed value for this position of frame
N / A	11	Hex 30	DB	Fixed value for this position of frame
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
N / A	15	Hex XXXX	DW	Don't care value for this position of frame
NACKREAS	17	?	DB	Reason why frame was not acknowledged
N / A	18	Hex XX	DB	Don't-care value for this position of frame
N / A	19	Hex 0401	DW	Fixed value for this position of frame
N / A	1B	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1D	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1F	Hex 10XX	DW	Fixed value for this position of frame
N / A	21	Hex 0000	DW	Fixed value for this position of frame

Figure 2-10 (Part 1 of 2). Name Claim Response Frame

Field Name	Offset (Hex)	Length	Type	Description
DNAME	23	16 DUP(?)	DB	ASCII name for destination
DNCID	33	?	DW	Number used to determine the session for the frame or to identify the datagram
CRC	35	?	DD	Check byte
EDF	39	Hex 7E	DB	End-of-frame byte

Figure 2-10 (Part 2 of 2). Name Claim Response Frame

Name Query Cancel Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address This is the next node address (initiator's address)
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 50XX	DW	Fixed value for this position of frame
N / A	11	Hex A0	DB	Fixed value for this position of frame
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept.
CONNID	13	?	DW	Field ID for frame. Choose a new CONNID value on the first transmission or repeat the previous CONNID on retransmissions.
N / A	15	Hex 0202	DW	Fixed value for this position of frame
N / A	17	Hex XXXX	DW	Don't-care value for this position of frame
N / A	19	Hex 0100	DW	Fixed value for this position of frame
N / A	1B	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1D	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1F	Hex 10XX	DW	Fixed value for this position of frame
N / A	21	Hex XX10	DW	Fixed value for this position of frame

Figure 2-11 (Part 1 of 2). Name Query Cancel Frame

Field Name	Offset (Hex)	Length	Type	Description
DNAME	23	16 DUP(?)	DB	ASCII name for destination
SNAME	33	16 DUP(?)	DB	ASCII name for source
PNCID	43	?	DW	See definition
DID	45	?	DW	Number that is incremented by 1 for each frame
SNCID	47	?	DW	Field ID for frame
DNODEID	49	?	DW	Low address
	4B	?	DW	Mid address
	4D	?	DW	High address
SNODEID	4F	?	DW	Low address
	51	?	DW	Mid address
	53	?	DW	High address
PNODEID	55	?	DW	Low address
	57	?	DW	Mid address
	59	?	DW	High address
CRC	5B	?	DD	Check byte
EDF	5F	Hex 7E	DB	End-of-frame byte

Figure 2-11 (Part 2 of 2). Name Query Cancel Frame

Session Request Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address This is the next node address (initiator's address)
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 40XX	DW	Fixed value for this position of frame
POLL	11	Hex 0?	DB	Hex 00 to hex 07 means no poll. Hex 08 to hex 0F means to send a return frame (poll).
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
SEQ	15	?	DB	Session frame sequence number
ACK	16	?	DB	Number that includes number + 1 modulo 256 of last correctly received frame
N / A	17	Hex 0001	DW	Fixed value for this position of frame
PKTSIZE	19	?	DW	Frame size that can be accepted from the remote node
N / A	1B	Hex 0000	DW	Fixed value for this position of frame
N / A	1D	Hex 1010	DW	Fixed value for this position of frame
SNAME	1F	16 DUP(?)	DB	ASCII name for source

Figure 2-12 (Part 1 of 2). Session Request Frame

Field Name	Offset (Hex)	Length	Type	Description
DNAME	2F	16 DUP(?)	DB	ASCII name for destination
DNCID	3F	?	DW	Number used to determine the session for the frame or to identify the datagram
CRC	41	?	DD	Check byte
EDF	45	Hex 7E	DB	End-of-frame byte

Figure 2-12 (Part 2 of 2). Session Request Frame

Session Accept Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address This is the next node address (initiator's address)
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 40XX	DW	Fixed value for this position of frame
POLL	11	Hex 0?	DB	Hex 00 to hex 07 means no poll. Hex 08 to hex 0F means to send a return frame (poll).
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
SEQ	15	?	DB	Session frame sequence number
ACK	16	?	DB	Number that includes number + 1 modulo 256 of last correctly received frame
N / A	17	Hex 0002	DW	Fixed value for this position of frame
PKTSIZE	19	?	DW	Frame size that can be accepted from the remote node
DNCID	1B	?	DW	Number used to determine the session for the frame or to identify the datagram
CRC	1D	?	DD	Check byte
EDF	22	Hex 7E	DB	End-of-frame byte

Figure 2-13. Session Accept Frame

Session Reject Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01	?	DW	Low address
	03	?	DW	Mid address
	05	?	DW	High address
SADDR	07	?	DW	Low address
	09	?	DW	Mid address
	0B	?	DW	High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 40XX	DW	Fixed value for this position of frame
POLL	11	Hex 0?	DB	Hex 00 to hex 07 means no poll. Hex 08 to hex 0F means to send a return frame (poll).
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
SEQ	15	?	DB	Session frame sequence number
ACK	16	?	DB	Number that includes number + 1 modulo 256 of last correctly received frame
N / A	17	Hex 03	DB	Fixed value for this position of frame
RVAL	18	?	DB	Indication of why the session was rejected
DNCID	19	?	DW	Number used to determine the session for the frame or to identify the datagram
CRC	1B	?	DD	Check byte
EDF	1F	Hex 7E	DB	End-of-frame byte

Figure 2-14. Session Reject Frame

ACK Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address This is the next node address (initiator's address)
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 4000	DW	Fixed value for this position of frame
POLL	11	Hex 4?	DB	Hex 40 to hex 47 means no poll. Hex 48 to hex 4F means to send a return frame (poll).
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
SEQ	15	?	DB	Session frame sequence number
ACK	16	?	DB	Number that includes number +1 modulo 256 of last correctly received frame
N / A	17	Hex XX	DB	Don't care value for this position of frame
DNCID	18	?	DW	Number used to determine the session for the frame or to identify the datagram
CRC	1A	?	DD	Check byte
EDF	1E	Hex 7E	DB	End-of-frame byte

Figure 2-15. ACK Frame

NACK Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 4000	DW	Fixed value for this position of frame
POLL	11	Hex 5?	DB	Hex 50 to hex 57 means no poll. Hex 58 to hex 5F means to send a return frame (poll).
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
SEQ	15	?	DB	Session frame sequence number
ACK	16	?	DB	Number that includes number + 1 modulo 256 of last correctly received frame
NACKREAS	17	?	DB	Reason why the frame was not received
DNCID	18	?	DW	Number used to determine the session for the frame or to identify the datagram
CRC	1A	?	DD	Check byte
EDF	1E	Hex 7E	DB	End-of-frame byte

Figure 2-16. NACK Frame

Close Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address This is the next node address (initiator's address)
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 4000	DW	Fixed value for this position of frame
POLL	11	Hex 6?	DB	Hex 60 to hex 67 means no poll. Hex 68 to hex 6F means to send a return frame (poll).
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
SEQ	15	?	DB	Session frame sequence number
ACK	16	?	DB	Number that includes number + 1 modulo 256 of last correctly received frame
N / A	17	Hex XX	DB	Fixed value for this position of the frame
DNCID	18	?	DW	Number used to determine the session for the frame or to identify the datagram
CRC	1A	?	DD	Check byte
EDF	1E	Hex 7E	DB	End-of-frame byte

Figure 2-17. Close Frame

Closed Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address This is the next node address (initiator's address)
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 4000	DW	Fixed value for this position of frame
N / A	11	Hex 70	DB	Fixed value for this position of frame
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
SEQ	15	?	DB	Session frame sequence number
ACK	16	?	DB	Number that includes number +1 modulo 256 of last correctly received frame
CLREAS	17	?	DB	Reason why connection closed
DNCID	18	?	DW	Number used to determine the session for the frame or to identify the datagram
CRC	1A	?	DD	Check byte
EDF	1E	Hex 7E	DB	End-of-frame byte

Figure 2-18. Closed Frame

Data Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address This is the next node address (initiator's address)
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 40XX	DW	Fixed value for this position of frame
POLL	11	Hex 0?	DB	Hex 00 to hex 07 means no poll. Hex 08 to hex 0F means to send a return frame (poll).
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
SEQ	15	?	DB	Session frame sequence number
ACK	16	?	DB	Number that includes number + 1 modulo 256 of last correctly received frame
EOM	17	Hex ?0	DB	End-of-message field. Hex (80-F0) equals end-of-message.
DATA	18	?? DUP(?)	DB	Variable length field
DNCID	XX	?	DW	Number used to determine the session for the frame or to identify the datagram
CRC	XX	?	DD	Check byte
EDF	XX	Hex 7E	DB	End-of-frame byte

Figure 2-19. Data Frame

Datagram Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01	?	DW	Low address
	03	?	DW	Mid address
	05	?	DW	High address
SADDR	07	?	DW	Low address
	09	?	DW	Mid address
	0B	?	DW	High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 51XX	DW	Fixed value for this position of frame
N / A	11	Hex 0100	DW	Fixed value for this position of frame
N / A	13	Hex 0001	DW	Fixed value for this position of frame
N / A	15	Hex 1010	DW	Fixed value for this position of frame
N / A	17	Hex 0000	DW	Fixed value for this position of frame
SNAME	19	16 DUP(?)	DB	ASCII name for source
DNAME	29	16 DUP(?)	DB	ASCII name for destination
DATA	39	?? DUP(?)	DB	Variable length field
PNCID	XX	?	DW	Equal to SNCID
DID	XX	?	DW	Number that is incremented by 1 for each frame
SNCID	XX	Hex FFFE	DW	Field ID for frame
DNODEID	XX	?	DW	Low address
	XX	?	DW	Mid address
	XX	?	DW	High address
SNODEID	XX	?	DW	Low address
	XX	?	DW	Mid address
	XX	?	DW	High address
PNODEID	XX	?	DW	Low address
	XX	?	DW	Mid address
	XX	?	DW	High address

Figure 2-20 (Part 1 of 2). Datagram Frame

Field Name	Offset (Hex)	Length	Type	Description
CRC	XX	?	DD	Check byte
EDF	XX	Hex 7E	DB	End-of-frame byte

Figure 2-20 (Part 2 of 2). Datagram Frame

Get Status Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 51XX	DW	Fixed value for this position of frame
N / A	11	Hex 0300	DW	Fixed value for this position of frame
N / A	13	Hex 0003	DW	Fixed value for this position of frame
CMDRESP	15	Hex ?2	DB	Hex (02-72) = command Hex (82-F2) = response
N / A	16	Hex XX	DB	Fixed value for this position of frame
N / A	17	Hex 8001	DW	Fixed value for this position of frame
N / A	19	Hex 8001	DW	Fixed value for this position of frame
TRANSID	1B	?	DW	ID for status and status response frame
N / A	1D	Hex 10	DB	Fixed value for this position of frame
DNAME	1E	16 DUP(?)	DB	ASCII name for destination
N / A	2E	Hex 00	DB	Fixed value for this position of frame
PNCID	2F	Hex FFFE	DW	Equal to SNCID
DID	31	?	DW	Number that is incremented by 1 for each frame
SNCID	33	Hex FFFE	DW	Field ID for frame

Figure 2-21 (Part 1 of 2). Get Status Frame

Field Name	Offset (Hex)	Length	Type	Description
DNODEID	35	?	DW	Low address
	37	?	DW	Mid address
	39	?	DW	High address
SNODEID	3B	?	DW	Low address
	3D	?	DW	Mid address
	3F	?	DW	High address
PNODEID	41	?	DW	Low address
	43	?	DW	Mid address
	45	?	DW	High address
CRC	47	?	DD	Check byte
EDF	4B	Hex 7E	DB	End-of-frame byte

Figure 2-21 (Part 2 of 2). Get Status Frame

Status Response Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01	?	DW	Low address
	03	?	DW	Mid address
	05	?	DW	High address
SADDR	07	?	DW	Low address
	09	?	DW	Mid address
	0B	?	DW	High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 51XX	DW	Fixed value for this position of frame
N / A	11	Hex 0300	DW	Fixed value for this position of frame
N / A	13	Hex 0003	DW	Fixed value for this position of frame
N / A	15	Hex 82	DB	Fixed value for this position of frame
N / A	16	Hex 00	DB	Fixed value for this position of frame
N / A	17	Hex 8001	DW	Fixed value for this position of frame
N / A	19	Hex 8001	DW	Fixed value for this position of frame
TRANSID	1B	?	DW	The transaction identification number
N / A	1D	Hex 10	DB	Fixed value for this position of frame
DNAME	1E	16 DUP(?)	DB	ASCII name for destination
N / A	2E	Hex 00	DB	Fixed value for this position of frame
STATLEN	2F	?	DW	Equal to the offset value of NRALIAS minus hex 12
DNODEID	31	?	DW	Low address
	33	?	DW	Mid address
	35	?	DW	High address

Figure 2-22 (Part 1 of 4). Status Response Frame

Field Name	Offset (Hex)	Length	Type	Description
JUMPSTAT	37	?	DB	Two high-order bits, indicating the position of jumpers W1 and W2. When set to 1, the jumpers are in place.
SELFTEST	38	?	DB	The result of the node's self-test
SWVERSION	39	?	DW	The software version used by the node
REPORTPD	3B	?	DW	The time in minutes since the last hardware reset
CRCERRS	3D	?	DW	Number of properly aligned frames received with a CRC error
ALGNERRS	3F	?	DW	Number of misaligned frames received with a CRC error
COLLISIONS	41	?	DW	Number of collisions encountered on transmission frames
XMITABRTS	43	Hex 0000	DW	Number of transmitted frames that transmitted unsuccessfully
XMITMSGS	45	?	DD	Number of frames successfully transmitted by the Link level
RECVMSGS	49	?	DD	Number of successfully received frames
REXMITCNT	4D	?	DW	Number of frames retransmitted
NORESOURCES	4F	?	DW	The number of frames discarded because of a lack of memory resources
N / A	51	Hex XXXX	DW	Don't-care value for this position of frame
N / A	53	Hex XXXX	DW	Don't-care value for this position of frame
N / A	55	Hex XXXX	DW	Don't-care value for this position of frame
N / A	57	Hex XXXX	DW	Don't-care value for this position of frame

Figure 2-22 (Part 2 of 4). Status Response Frame

Field Name	Offset (Hex)	Length	Type	Description
N / A	59	Hex XXXX	DW	Don't-care value for this position of frame
N / A	5B	Hex XXXX	DW	Don't-care value for this position of frame
N / A	5D	Hex XXXX	DW	Don't-care value for this position of frame
N / A	5F	Hex XXXX	DW	Don't-care value for this position of frame
N / A	61	Hex XXXX	DW	Don't-care value for this position of frame
N / A	63	Hex XXXX	DW	Don't-care value for this position of frame
N / A	65	Hex XXXX	DW	Don't-care value for this position of frame
N / A	67	Hex XXXX	DW	Don't-care value for this position of frame
N / A	69	Hex XXXX	DW	Don't-care value for this position of frame
NRALIAS	6B	?	DB	Number of alias names to follow in the frame. (The next three fields, ALIASNAME, ALIASNR, and ALIASTAT, are repeated in sequence with the number given in the NRALIAS field. This field's offset can be calculated by using the STATLEN field offset and adding hex 12.
ALIASNAME	6C	16 DUP(?)	DB	Alias name for 16 bytes
ALIASNR	7C	?	DB	Number assigned to the alias name
ALIASTAT	7D	?	DB	The status of the name specified in the ALIASNAME field
O	XX	O	O	
O	XX	O	O	
O	XX	O	O	

Figure 2-22 (Part 3 of 4). Status Response Frame

Field Name	Offset (Hex)	Length	Type	Description
ALIASNAME	XX	16 DUP(?)	DB	Alias name for 16 bytes
ALIASNR	XX	?	DB	Number assigned to the alias name
ALIASTAT	XX	?	DB	The status of the name specified in the ALIASNAME field
PNCID	XX	Hex FFFE	DW	See definition
DID	XX	?	DW	Number that is incremented by 1 for each frame
SNCID	XX	Hex FFFE	DW	Field ID for frame
DNODEID	XX	?	DW	Low address
	XX	?	DW	Mid address
	XX	?	DW	High address
SNODEID	XX	?	DW	Low address
	XX	?	DW	Mid address
	XX	?	DW	High address
PNODEID	XX	?	DW	Low address
	XX	?	DW	Mid address
	XX	?	DW	High address
CRC	XX	?	DD	Check byte
EDF	5F	Hex 7E	DB	End-of-frame byte

Figure 2-22 (Part 4 of 4). Status Response Frame

Abort Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address This is the next node address (initiator's address).
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 40XX	DW	Fixed value for this position of frame
POLL	11	Hex 9?	DB	Hex (90-97) means no poll. Hex (98-9F) means to send a return frame.
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
SEQ	15	?	DB	Session frame sequence number
ACK	16	?	DB	Number that includes number + 1 modulo 256 of last correctly received frame
N / A	17	Hex XX	DB	Don't-care value for this position of frame
DNCID	18	?	DW	Number used to determine the session for the frame or to identify the datagram
CRC	1A	?	DD	Check byte
EDF	1E	Hex 7E	DB	End-of-frame byte

Figure 2-23. Abort Frame

Self-Test Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01	?	DW	Low address
	03	?	DW	Mid address
	05	?	DW	High address
SADDR	07	?	DW	Low address
	09	?	DW	Mid address
	0B	?	DW	High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex F0XX	DW	Fixed value for this position of frame
CRC	11	?	DD	Check byte
EDF	15	Hex 7E	DB	End-of-frame byte

Figure 2-24. Self-Test Frame

Ident Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 51XX	DW	Fixed value for this position of frame
N / A	11	Hex 0300	DW	Fixed value for this position of frame
N / A	13	Hex 0003	DW	Fixed value for this position of frame
CMDRESP	15	Hex ?1	DB	Hex 41 = command Hex C1 = response
N / A	16	Hex 00	DB	Fixed value for this position of frame
N / A	17	Hex 8001	DW	Fixed value for this position of frame
N / A	19	Hex 8001	DW	Fixed value for this position of frame
TRANSID	1B	Hex 0000	DW	ID for status and status response frame
N / A	1D	Hex 10	DB	Fixed value for this position of frame
DNAME	1E	16 DUP(?)	DB	ASCII name for destination
N / A	2E	Hex 00	DB	Fixed value for this position of frame
NODEIDMASK	2F 31 33	? ? ?	DW DW DW	Mask value Mask value Mask value
SNAME	33	16 DUP(?)	DB	ASCII name for source

Figure 2-25 (Part 1 of 2). Ident Frame

Field Name	Offset (Hex)	Length	Type	Description
NODEIDMATCH	35	?	DW	Match value
	37	?	DW	
	39	?	DW	
PNCID	3B	Hex FFFE	DW	Equal to SNCID
DID	3D	?	DW	Number that is incremented by 1 for each frame
SNCID	3F	Hex FFFE	DW	Field ID for frame
DNODEID	41	?	DW	Low address
	43	?	DW	Mid address
	45	?	DW	High address
SNODEID	47	?	DW	Low address
	49	?	DW	Mid address
	4B	?	DW	High address
PNODEID	4D	?	DW	Low address
	4F	?	DW	Mid address
	51	?	DW	High address
CRC	53	?	DD	Check byte
EDF	57	Hex 7E	DB	End-of-frame byte

Figure 2-25 (Part 2 of 2). Ident Frame

Section 3. PC Network Protocol Driver

The IBM PC Network Protocol Driver is a loadable software module that allows the PC Network adapters to use the NETBIOS on the network. The Protocol Driver implementation of NETBIOS supports up to 64 concurrent duplex sessions, 62 names, and 32 NCB commands.

NCB Return Codes

The Protocol Driver generates the following additional NETBIOS return codes.

1A **Incompatible remote device**

Cause: Unexpected protocol packet received.

Action: Verify that all nodes on the network agree with the network protocols.

41 **Hot carrier detected**

Cause: The adapter detected a hot carrier on the network.

Action: Perform the node diagnostic tests.

42 **Hot carrier sent**

Cause: The adapter sent a hot carrier on the network.

Action: Perform the node diagnostic tests.

43 **No carrier detected**

Cause: No carrier was detected by the adapter.

Action: Perform the node diagnostic tests.

44 to 4D Unusual network condition

Cause: The NETBIOS detected an unusual condition on the network.

Action: Retry or reset the command. If you receive the return code again, perform the node diagnostic tests.

The following summarizes the commands that use these return codes.

Return Code	NCB Command	Type
1A	CALL	Final Return Code
	LISTEN	Final Return Code

Figure 3-1. PC Network Adapter NCB Return Codes

Protocol Driver Commands

The following commands are specific to the Protocol Driver.

Status (Adapter Status) Command

Data that is returned in the data buffer field contains the following information.

Unit identification number—6 bytes: This number is part of the permanent node name. The unit identification number is represented as follows:

Byte 0: Low word, low byte
Byte 1: Low word, high byte
Byte 2: Middle word, low byte
Byte 3: Middle word, high byte
Byte 4: High word, low byte
Byte 5: High word, high byte.

External option status—1 byte: The status of the adapter options is represented below:

PC Network Adapter II

Bit	Function
7	Adapter Present
6	Driver Loaded
5	Reserved
4	Interrupt level 1 = Interrupt level 3 0 = Interrupt level 2
3	FIFO address 1 = hex DC00 0 = hex CC00
2	Error Detected
1	Remote Program Load Status 1 = On 0 = Off
0	Adapter good and initialized

Figure 3-2. PC Network Adapter II Option Byte

PC Network Adapter II/A

Bit	Function
7	Adapter Present
6	Driver Loaded
5	FIFO Address high bit 11 = hex DE00 10 = hex D600
4	Interrupt level 1 = Interrupt level 3 0 = Interrupt level 2
3	FIFO address 01 = hex CE00 00 = hex C600
2	Error Detected
1	Remote Program Load Status 1 = On 0 = Off
0	Adapter good and initialized

Figure 3-3. PC Network Adapter II/A Option Byte

Software version—2 bytes: The software version of the protocol layers are represented as follows:

Byte 0: Major version number
Byte 1: Minor version number

Traffic and error statistics—48 bytes:

1. Duration of reporting period (in minutes)—2 bytes. After the counter reaches a value of hex 0FFFF, it resets to 0.
2. The number of CRC errors received—2 bytes. After the counters reach a value of hex 0FFFF, they do not increment further.[1]
3. The number of alignment errors received—2 bytes. After the counters reach a value of hex 0FFFF, they do not increment further.[1]
4. The number of collisions encountered—2 bytes. After the counter reaches a value of hex 0FFFF, it resets to 0.
5. The number of unsuccessful transmissions—2 bytes. After the counter reaches a value of hex 0FFFF, it resets to 0.[1]

---

[1] See Intel's Microcommunications Handbook for further information.

6. Number of successfully transmitted frames—4 bytes. After the counter reaches a value of hex 0FFFFFFFF, it resets to 0.
7. Number of successfully received frames—4 bytes. After the counter reaches a value of hex 0FFFFFFFF, it resets to 0.
8. Number of retransmissions—2 bytes. After the counter reaches a value of hex 0FFFF, it resets to 0.
9. Number of times the receiver exhausted its resources—2 bytes. After the counters reach a value of hex 0FFFF, they do not increment further.[2]

Adapter resource statistics:

1. Reserved for internal use—8 bytes.
2. Number of NCBs currently available for use—2 bytes.
3. Maximum number of NCBs supported by the adapter as configured by the Reset command—2 bytes.
4. Maximum number of NCBs possible for the adapter to support—2 bytes.
5. Reserved for internal use—4 bytes.
6. Pending sessions—2 bytes. A pending session is either a CALL-pending, a LISTEN-pending, a session established, a session ended abnormally, HANG UP-pending, or HANG UP-complete.
7. Maximum number of sessions supported by the adapter as configured by Reset command—2 bytes.
8. Maximum number of sessions possible for the adapter to support—2 bytes
9. Maximum session data frame size—2 bytes.

Quantity of names in the local name table—2 bytes: This number is the quantity of names present in the local name table.

Local name table—18 bytes per user name in table: The first 16 bytes of each entry represent the name, and the last 2 bytes represent the

---

[2] See Intel's Microcommunications Handbook for further information.

name status. This first byte is equal to the name number. The second byte denotes the status when it is masked with a hex 87. The mask is used to get the most-significant bit and the last 3 bits of the byte. The other bits are reserved and can have nonzero values.

- NXXXX000 = Trying to register a name
- NXXXX100 = A registered name
- NXXXX101 = A deregistered name
- NXXXX110 = A detected duplicate name
- NXXXX111 = A detected duplicate name with deregister pending.

  Where:
  - X = Reserved bit
  - N = 0 The name is a unique name
  - N = 1 The name is a group name.

Call Command

The Protocol Driver implementation does not support the return code of 0B (command canceled) for a Call command.

Cancel Command

The following commands are not valid to CANCEL: ADD NAME, ADD GROUP NAME, DELETE NAME, SEND DATAGRAM, SEND BROADCAST DATAGRAM, SESSION STATUS, RESET, and CANCEL; the command, CALL, will never be canceled.

Unlink Command

The Protocol Driver implementation does not support this command. If an Unlink command is issued, a hex 00 will be returned.

PC Network Protocol Driver Pseudocode for NCB Commands

The following is the pseudocode for the NCB commands.

RESET

```
PROCEDURE Reset (command_block);
        {set configuration parameters}
BEGIN
IF either of the two configuration
    parameters is equal to zero THEN;
    set configuration parameter to zero
    {6 sessions, and 12 commands}
ELSE
IF either of the two configuration parameters is greater
     than maximum value
    set configuration parameter to maximum value
    BEGIN
    set configuration parameters in configuration table;
    return command_completed status to user;
    END
END;
```

STATUS

```
PROCEDURE Status (command_block);
    {get LANA configuration parameters and status}
BEGIN
IF buffer length is invalid THEN;
    return invalid_buffer_length status to user;
ELSE
    IF name in conflict THEN;
        return name_conflict status to user;
    ELSE
        IF name is deregistered THEN;
           return invalid_name to user;
    ELSE
        IF name is a local name or name is * THEN;
        BEGIN
        get configuration parameters and status of local LANA;
        return configuration parameters and status;
        return actual length of configuration
            parameter and status;
        IF size of user buffer is smaller than the configuration
            parameter and status THEN;
            return message_incomplete status to user;
        ELSE
            return command_completed status to user;
        END
    ELSE
        BEGIN
        send status_request frame to remote node;
        wait for response from remote node;
            {remote node send status_response frame}
        IF status_response frame is received within the time-out
            interval THEN;
            BEGIN
            extract configuration parameters and
                status of remote LANA;
            return configuration parameters and status;
            return actual length of configuration
                parameter and status;
            IF size of user buffer is smaller than the configuration
                parameter and status THEN;
                return message_incomplete status to user;
            ELSE
                return command_completed status to user;
            END
        ELSE
            return command_timed_out status to user;
        END
END;
```

ADD NAME

```
PROCEDURE Add_Name (command_block);
        {request local registration of name}
BEGIN
IF name begins with * or null THEN;
    return invalid_name status to user;
ELSE
    BEGIN
    search for name in local name table;
    IF name is found THEN;
        BEGIN
            IF name in conflict THEN;
                return name_conflict status to user;
            ELSE
              IF name is deregistered THEN;
                 return invalid_name status to user
            ELSE return existing name_number to user;
            return duplicate_name status to user;
        END
    ELSE
        IF local name table is full THEN;
            return name_table_full status to user;
        ELSE
            BEGIN
              IF no resource for transmission THEN;
                 return no_resource_available status to user
            REPEAT
            ELSE
                BEGIN
                broadcast name_claim frame to network;
                wait for response from remote node in network;
                {all remote nodes search for name
                in their local name tables and send
                name_claim_response frame if found}
                END
            UNTIL name_claim_response frame is received
                or number of times to broadcast is reached;
            IF name_claim_response frame is not received THEN;
                BEGIN
                enter name to local name table;
                return name_number to user;
                return command_completed status to user;
                END
            ELSE
                return name_in_use status to user;
            END
    END
END;
```

ADD GROUP NAME

```
PROCEDURE Add_Group_Name (command_block);
       {request local registration of name}
BEGIN
IF name begins with * or null THEN;
    return invalid_name status to user;
ELSE
    BEGIN
    search for name in local name table;
    IF name is found THEN;
        BEGIN
            IF name in conflict THEN;
                return name_conflict status to user;
            ELSE
               IF name is deregistered THEN;
                 return invalid_name status to user;
            ELSE
            return existing name_number to user;
            return duplicate_name status to user;
        END
    ELSE
        IF local name table is full THEN;
            return name_table_full status to user;
        ELSE
            BEGIN
            IF no resource available for transmission THEN;
               return no_resource_available status to user
            ELSE
            REPEAT
                BEGIN
                broadcast add_group_name_claim frame to network;
                wait for response from remote node in network;
                {all remote nodes search for name in their local name
                tables and send name_claim_response
                frame if found}
                END
            UNTIL name_claim_response frame is received or
                number of times to broadcast is reached;
            IF name_claim_response frame is not
            received THEN;
                BEGIN
                enter name to local name table;
                return name_number to user;
                return command_completed status to user;
                END
            ELSE
                return name_in_use status to user;
            END
    END
END;
```

DELETE NAME

```
PROCEDURE Delete_Name (command_block);
        {request local deregistration of name}
BEGIN
IF name begins with * or null THEN;
    return invalid_name status to user;
ELSE
    BEGIN
    search for name in local name table;
    IF name is found THEN;
        BEGIN
        check for pending nonactive session commands;
        IF nonactive session command is found THEN;
            terminate the nonactive session command;
           check the session count in table entry;
        IF session count is zero THEN;
            BEGIN
            remove name from local name table;
            return command_complete status to user;
            END
        ELSE
            BEGIN
            change status of name to deregistered;
                {name will be removed from local name
                table when session count reaches zero}
            return command_completed_name_has_
                active_session status to user;
            END
        END
    ELSE
        return invalid_name status to user;
    END
END;
```

CALL

```
PROCEDURE Call (command_block);
    {open user session with remote name using name supplied}
BEGIN
IF local resource is not available THEN;
    return no_resource_available status to user;
ELSE
    BEGIN
    search for name in local name table;
    IF name is not found or name is deregistered THEN;
        return invalid_name status to user;
    ELSE
    IF name is marked "conflict detected" THEN;
        return name_conflict_detected status to user;
    ELSE
        IF local session table is full THEN;
            return local_session_table_full status to user;
        ELSE
            BEGIN
            search for remote name in local name table;
                BEGIN
                REPEAT
                    BEGIN
                    IF name is not found THEN
                        broadcast name_query frame to network;
                    ELSE loopback name_query frame;
                        wait for response from remote node or
                        time-out;
                        {all remote nodes search for name
                        in their local name tables and send
                        name_query_response frame if found}
                    END
                UNTIL name_query_response frame is received
                    or number of times to broadcast is
                    reached;
                IF name_query_response frame is not
                    received THEN;
                    return unknown_remote_name status to user;
                END
                BEGIN
                send session_request frame to destination
                    node;
                    {destination node search for pending
                    LISTEN command, if found, return
                    session_accept frame, else, return
                    session_reject frame}
                IF network error THEN;
                    return session_aborted to user;
                IF response from destination is received
                    within the time-out interval THEN;
                    CASE response OF
```

```
                    session_accept frame:
                        BEGIN
                        set session_established indicator
                            in session table;
                        return local_session_number
                            to user;
                        return command_completed status
                            to user;
                        END
                        {session established}
                    session_reject frame:
                        return session_open_rejected status
                            to user;
                END {case}
            ELSE
                return command_timed_out status to user;
            END
        END
    END
END;
```

LISTEN

```
PROCEDURE Listen (command_block);
    {open user session with remote name, using
    name supplied}
BEGIN
IF local resource is not available THEN;
    return no_resource_available status to user;
ELSE
    BEGIN
    search for name in local name table;
    IF name is not found or name is deregistered THEN;
        return invalid_name status to user;
    ELSE
        IF name is marked "conflict detected" THEN
            return name_conflict_detected status to user;
        ELSE
            IF local session table is full THEN;
                return local_session_table_full status to user;
            ELSE
                BEGIN
                REPEAT
                    BEGIN
                    wait for session_request frame
                        or name_conflict_detection;
                    IF LISTEN specific is specified THEN;
                        check source of session_request frame;
                        IF source of session_request frame is
                            same as remote name THEN;
                            set session_request_completed indicator;
                            {LISTEN specific satisfied}
                        ELSE
                            reset session_request_completed
                                indicator;
                            {LISTEN not satisfied, continue
                            to wait}
                    ELSE
                        IF LISTEN any specified THEN
                        set session_request_completed indicator;
                        {LISTEN ANY satisfied}
                    END

                UNTIL session_request_completed indicator is set
                    or name_conflict_detected;
                IF name_conflict_detected for outstanding
                    LISTEN THEN
                ELSE IF network error THEN;
                    return session_aborted status to user;
                     ELSE
                         send session_accept frame to source;
                         wait for first frame on session;
                         set session_established
```

```
                    indicator in session table;
                return source of session_request
                    frame to user;
                return local_session_number to user;
                return command_completed status to user;
                {session established}
            END
        END
END;
```

HANG UP

```
PROCEDURE Hang Up (command_block);
    {close user session indicated by
    local_session_number}
BEGIN
IF local session number is invalid THEN;
    return invalid_session_number status to user;
ELSE
    IF session is already closed and session_closed status has
        not been reported THEN;
        return session_closed status to user;
    ELSE
        IF session is already aborted and session_aborted
            status has not been reported THEN;
            return session_aborted status to user;
        ELSE
            BEGIN
            REPEAT
                IF a RECEIVE command is pending THEN;
                    terminate RECEIVE command with a
                    session_closed status to user;
            UNTIL all pending RECEIVE commands
                are terminated;
            REPEAT
                IF a SEND or CHAIN SEND command
                    is pending THEN;
                    wait until the SEND, CHAIN SEND, or
                        HANG UP has completed or
                        timed out;
                IF the SEND command was timed out THEN
                    abort the session;
            UNTIL all pending SEND and CHAIN SEND
                commands are completed or timed out;
            send close frame to destination node;
            wait for close frame from destination node or
                close time-out;
                {destination node close the session
                and send close frame}
            IF close frame is received before the close
                time-out interval THEN;
                BEGIN
                close the session;
                return command_completed status to user;
                END
            ELSE
                BEGIN
                abort the session;
                return session_aborted status to user;
                END
            IF RECEIVE ANY to name command is pending THEN;
                terminate RECEIVE ANY to name command with
```

```
            session_closed or session_aborted status;
        ELSE
            IF a RECEIVE ANY to any name command
            is pending THEN
                terminate the RECEIVE ANY to any name command
                with session_closed or
                session_aborted status;
        END
END;
```

SEND

```
PROCEDURE Send (command_block);
    {send data through user session as indicated by
    local_session_number}
BEGIN
IF local session number is invalid THEN;
    return invalid_local_session_number
    status to user;
ELSE
    IF session is closed and session_closed status has not
        been reported THEN;
        return session_closed status to user;
    ELSE
        IF session is aborted and session_aborted status has
            not been reported THEN;
            return session_aborted status to user;
        ELSE
            BEGIN
            send session data frame(s) to destination node;
            wait for ack frame(s) from destination node
                or for time-out;
            IF session data is sent successfully within the
                time-out interval for session send THEN;
                return command_completed status to user;
            ELSE
                BEGIN
                abort the session;
                return command_timed_out status to user;
                END
            END
END;
```

CHAIN SEND

```
PROCEDURE Chain Send (command_block);
    {send data through user session as indicated by
    local_session_number}
BEGIN
IF local session number is invalid THEN;
    return invalid_local_session_number status to user;
ELSE
    IF session is closed and session_closed status
        has not been reported THEN;
        return session_closed status to user;
    ELSE
        IF session is aborted and session_aborted
            status has not been reported THEN;
            return session_aborted status to user;
        ELSE
            BEGIN
            send session data frame(s) to destination node;
            wait for ack frame(s) from destination node
                or for time-out;
            IF session data is sent successfully within the
                time-out interval for session send THEN;
                return command_completed status to user;
            ELSE
                BEGIN
                abort the session;
                return command_timed_out status to user;
                END
            END
END;
```

RECEIVE

```
PROCEDURE Receive (command_block);
    {receive data through user session as indicated by
    local_session_number}
BEGIN
IF local session number is invalid THEN;
    return invalid_local_session_number
        status to user;
ELSE
    IF session is closed and session_closed status has not
        been reported THEN;
        return session_closed status to user;
    ELSE
        IF session is aborted and session_aborted status has
            not been reported THEN;
            return session_aborted status to user;
        ELSE
            BEGIN
            wait for session message (data frame(s))
                from source node;
            IF session data is received within the time-out
                interval for session receive THEN;
                BEGIN
                send ack frames(s) to source node;
                transfer session data to user buffer
                    of appropriate length;
                return actual length of transfer to user;
                IF size of user buffer is smaller than
                    received session data THEN;
                    return message_incomplete status to user;
                ELSE
                    return command_completed status to user;
                END
            ELSE
                return command_timed_out status to user;
                {session data received}
            END
END;
```

RECEIVE ANY

```
PROCEDURE Receive_Any (command_block);
    {receive any data sent to the
    specified name_number}
BEGIN
IF name number is invalid or deregistered THEN;
    return invalid_name_number
        status to user;
ELSE
    IF session is closed and session_closed
        status has not been reported THEN;
        return session_closed status to user;
    ELSE
        IF session is aborted and session_aborted
            status has not been reported THEN;
            return session_aborted status to user;
        ELSE
            BEGIN
            REPEAT
                BEGIN
                wait for a session message (data frame(s));
                IF RECEIVE specific is specified THEN;
                    BEGIN
                    check recipient name in session message;
                    IF recipient name in session message is
                        same as local name THEN;
                        set receive_completed indicator;
                        {receive to specific name satisfied}
                    ELSE
                        reset receive_completed indicator;
                        {receive to specific not satisfied,
                        continue to wait}
                    END
                ELSE
                    set receive_completed indicator;
                    {receive to any name satisfied}
                END
            UNTIL receive_completed indicator is set;
            IF "conflict detected" error THEN;
                return name_conflict_detected status to user;
            ELSE
                send ack frame(s) to sending node;
                transfer session data to user buffer
                    of appropriate length;
                return actual length of transfer to user;
                return recipient name number to user;
                return local_session_number to user;
                IF size of user buffer is smaller than
                    received session data THEN;
```

```
                return message_incomplete status to user;
            ELSE
                return command_completed status to user;
                {session data received}
        END
END;
```

SESSION STATUS

```
PROCEDURE Session_Status (command_block);
    {obtain status of session indicated by name}
BEGIN
IF buffer length is invalid THEN;
    return invalid_buffer_length status to user;
ELSE
    BEGIN
        IF name in conflict THEN
            return name_conflict_detected status to user;
        IF name does not start with * THEN;
            IF name is registered search for name in local name table;
            THEN return number of pending sessions;
            return number of pending datagram receives;
            IF name or * is found THEN;
                BEGIN
                get session status in session table
                    for each session on the name or;
                    for all session if * ;
                    return session status to user;
                    return actual length of session status;
                IF size of user buffer is smaller than
                    size of session status data THEN;
                    return message_incomplete status to user;
                ELSE
                    return command_completed status to user;
                END
        ELSE
            return invalid_name status to user;
        END
END;
```

SEND DATAGRAM

```
PROCEDURE Send Datagram (command_block);
    {send datagram to remote node with the specified
    name registration}
BEGIN
IF buffer length is invalid THEN;
    return invalid_buffer_length status to user;
ELSE
    BEGIN
    search for name corresponding to name number
        in local name table;
    IF name in conflict THEN;
        return name_conflict status to user;
    ELSE
        IF registered name is not found THEN;
            return invalid_name_number status to user;
        ELSE
            BEGIN
            send datagram to destination node;
            return command_completed status to user;
            END
    END
END;
```

SEND BROADCAST DATAGRAM

```
PROCEDURE Send_Broadcast_Datagram (command_block);
    {send broadcast datagram to all nodes on the network}
BEGIN
IF buffer length is invalid THEN;
    return invalid_buffer_length status to user;
ELSE
    BEGIN
    search for name in local name table;
    IF name in conflict THEN;
        return name_conflict status to user;
    ELSE
        IF registered name is not found THEN;
            return invalid_name_number status to user;
        ELSE
            BEGIN
            broadcast datagram to all nodes on the network;
            return command_completed status to user;
            END
    END
END;
```

RECEIVE DATAGRAM

```
PROCEDURE Receive Datagram (command_block);
    {receive datagram from any node on the network}
BEGIN
search for name corresponding to name number
    in local name table;
IF registered name is not found THEN;
    return invalid_name_number status to user;
ELSE
    BEGIN
    REPEAT
        BEGIN
        wait for arrival of datagram;
        IF datagram receive specific is specified THEN;
            BEGIN
            check recipient name in datagram message;
            IF recipient name in datagram message is same
                as name specified by local name
                number THEN;
                set receive_completed indicator;
                {datagram receive to
                specific name satisfied}
            ELSE
                reset receive_completed indicator;
                {datagram receive not satisfied,
                continue to wait}
            END
        ELSE
            set receive_completed indicator;
            {datagram receive to any name satisfied}
        END
    UNTIL receive_completed indicator is set;
    IF "conflict detected" error THEN;
        return name_conflict_detected status to user;
    ELSE
        transfer datagram data to user buffer
            of appropriate length;
        return actual length of transfer to user;
        return local name_number to user;
        return sender's name to user;
        IF size of user buffer is smaller than
            received datagram THEN;
            return message_incomplete status to user;
            {data received, unable to
            transfer entire message}
        ELSE
            return command_completed status to user;
            {datagram received}
    END
END;
```

RECEIVE BROADCAST DATAGRAM

```
PROCEDURE Receive Broadcast Datagram (command_block);
    {receive broadcast datagram from
    any node in the network}
BEGIN
search for name corresponding to name number in
    local name table;
IF registered name is not found THEN;
    return invalid_name_number status to user;
ELSE
    wait for arrival of broadcast datagram;
    IF "conflict detected" error THEN;
        return name_conflict_detected status to user;
    ELSE
        transfer datagram to user buffer
            of appropriate length;
        return actual length of transfer to user;
        return sender's name to user;
        IF size of user buffer is smaller than received
            datagram THEN;
            return message_incomplete status to user;
            {broadcast datagram received, unable to return
            entire message}
        ELSE
            return command_completed status to user;
            {broadcast datagram received}
END;
```

PC Network Protocol Driver Frame Description

The following describes the protocol frames and their functions.

Field Definitions

The description of the fields within the protocol frame types follows.

Field	Meaning
ACK	This is an 8-bit field that includes the sequence number + 1 modulo 256 of the last correctly received frame.
ALGNERRS	This is a 16-bit field specifying the number of frames received with alignment errors.
ALIASNAME	This is a 16-byte field containing the name.
ALIASNR	This is an 8-bit field specifying the number that is assigned to a given name.
ALIASTAT	The low-order 4 bits specify the status of a name. See page 2-7 for the numbers that are returned in this field.
CLREAS	This is an 8-bit reason code indicating the reason for a connection being closed. CLREAS = hex 00 indicates a normal close.
CMDRESP	This is a 1-bit flag indicating whether the frame contains a command or a response to a command.
COLLISIONS	This is a 16-bit field specifying the number of transmitted frames that experienced collisions.
CONNID	This is a 16-bit identifier used to determine which session a frame is assigned to.
CRC	This is a 32-bit field containing the cyclic redundancy check for the frame according to the Autodin-II 32-bit polynomial generator.
CRCERRS	This is a 16-bit field specifying the number of CRC errors being reported.

DADDR This 48-bit field identifies the destination Link level address of the frame. A value of all binary 1's indicates a broadcast. All other adapter addresses will have the 16 highest bits set to 0. A value whose least-significant bits are hex FF indicates a group address of the form f(name).

DATA DATA is a variable length field containing user data.

DID DID is a 16-bit field that is incremented with each retransmission of a name query, name claim, and a get status frame.

DLEN This 16-bit field contains the length of the data field, in bytes, of the Link level frame.

DNAME This is a 16-bit field that identifies the name of a destination (ASCII characters).

DNCID This is a 16-bit field used with CONNID to determine the session for each frame or to identify a datagram.

DNODEID This is a 48-bit field indicating the Link level address of an intended destination.

EFD This is an 8-bit end-frame delimiter flag (hex 7E).

EOM This is a 1-bit end-of-message indicator. The bit marks the end of a user's logical message.

JUMPSTAT1 This is a 1-bit field indicating the status of jumper W1. When set to 1, jumper W1 is in place.

JUMPSTAT2 This is a 1-bit field indicating the status of jumper W2. When set to 1, jumper W2 is in place.

NACKREAS This is an 8-bit field indicating the reason why a frame was not successfully delivered. The following is a list of the codes that can be contained in this field

- Hex 00 — Standard NACK frame
- Hex 10 — Bad sequence number
- Hex 12 — Rejected by a higher level
- Hex 18 — NACK response to an unexpected open request
- Hex 20 — Incompatible RSP version

- Hex 22 – Bad service ID
- Hex 24 – Invalid RSP control information
- Hex 26 – No remote resources
- Hex 32 – Acceptable NACK frame
- Hex 34 – Name claim rejected
- Hex 35 – Name query rejected

NODEIDMASK This is a 6-byte mask used in a logical AND operation to get the destination node name.

NODEIDMATCH This is the 6-byte match value to use to request a response.

NRALIAS This is the 16-bit field specifying the number of names to follow.

PKTSIZE This is a 16-bit field indicating the maximum frame size that an initiating session node is willing to accept.

PNCID This is a 16-bit field used with CONNID to determine the network connection with which a frame is associated at a previous node, or to identify a datagram.

PNODEID This is a 48-bit field indicating the Link level address of the previous node.

POLL This is a 1-bit field which, when set to hex 08, indicates that a return frame should be generated back to the sender. Hex 00 indicates no poll.

RECVMSGS This is a 32-bit field specifying the number of frames that have been successfully received for a given period of time.

REPORTPD This is a 16-bit field specifying the period of time, in minutes, for the statistical data that was gathered.

RVAL This is an 8-bit value indicating the reason a requested session was rejected. RVAL = 3 indicates no matching Listen command. RVAL = 4 indicates an incompatible version.

SADDR This is a 48-bit field identifying the source Link level address of the frame.

SCONNID This is a 16-bit identifier used to determine the session number for a frame.

SEQ SEQ is an 8-bit session frame sequence number. It is incremented with each *new* data transmission.

SHORTFRAMES This is a 16-bit field specifying the number of short frames being reported.

SNAME This is a 16-bit field containing the name of a source node (specified in ASCII characters).

SNCID This is a 16-bit field used with CONNID to determine the network connection with which a frame is associated at a source node, or to identify a datagram.

SNODEID This is 48-bit field indicating the Link level address of a source node.

STD STD is an 8-bit start delimiter flag (hex 7E).

TRANSID This is a 16-bit field indicating the transaction ID for status and status response frames.

WIN WIN is a 4-bit field that indicates the number of frames beyond those already acknowledged that the sender is willing to accept.

XMITABRTS This is a 16-bit field specifying the number of transmitted frames that were ended abnormally.

XMITMSGS This is a 32-bit field specifying the number of frames that were successfully transmitted for a given period of time.

XXXX The Xs represent a "don't-care" value.

Frame Types, Formats, and Functions

The following describes the protocol frame types and their functions.

Note: A "?" refers to a variable value that has meaning to the particular field for that frame.

Name Query Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 50XX	DW	Fixed value for this position of frame
N / A	11	Hex 10	DB	Fixed value for this position of frame
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept
CONNID	13	?	DW	Field ID for frame. Choose a new CONNID value on the first transmission or repeat the previous CONNID on retransmissions
N / A	15	Hex 0202	DW	Fixed value for this position of frame
N / A	17	Hex XXXX	DW	Don't-care value for position of frame
N / A	19	Hex 0100	DW	Fixed value for this position of frame
N / A	1B	Hex XXXX	DW	Don't-care value for position of frame

Figure 3-4 (Part 1 of 2). Name Query Frame

Field Name	Offset (Hex)	Length	Type	Description
N / A	1D	Hex XXXX	DW	Don't-care value for position of frame
N / A	1F	Hex 10XX	DW	Fixed value for this position of frame
N / A	21	Hex XX10	DW	Fixed value for this position of frame
DNAME	23	16 DUP(?)	DB	ASCII name for destination
SNAME	33	16 DUP(?)	DB	ASCII name for source
PNCID	43	?	DW	See definition
DID	45	?	DW	Number that is incremented by 1 for each frame
SNCID	47	?	DW	Field ID for frame
DNODEID	49	?	DW	Low address
	4B	?	DW	Mid address
	4D	?	DW	High address
SNODEID	4F	?	DW	Low address
	51	?	DW	Mid address
	53	?	DW	High address
PNODEID	55	?	DW	Low address
	57	?	DW	Mid address
	59	?	DW	High address
CRC	5B	?	DD	Check byte
EDF	5F	Hex 7E	DB	End-of-frame byte

Figure 3-4 (Part 2 of 2). Name Query Frame

Name Claim and Name Claim Cancel Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01	?	DW	Low address
	03	?	DW	Mid address
	05	?	DW	High address
SADDR	07	?	DW	Low address
	09	?	DW	Mid address
	0B	?	DW	High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 50XX	DW	Fixed value for this position of frame
N / A	11	?	DB	Hex 10 = Name Claim frame Hex A0 = Name Claim cancel frame
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept
CONNID	13	?	DW	Field ID for frame. Choose a new CONNID value on the first transmission or repeat the previous CONNID on retransmissions.
N / A	15	Hex 0202	DW	Fixed value for this position of frame
N / A	17	Hex XXXX	DW	Don't-care value for this position of frame
N / A	19	Hex 0400	DW	Fixed value for this position of frame
N / A	1B	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1D	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1F	Hex 10XX	DW	Fixed value for this position of frame
N / A	21	Hex 0000	DW	Fixed value for this position of frame

Figure 3-5 (Part 1 of 2). Name Claim and Name Claim Cancel Frame

Field Name	Offset (Hex)	Length	Type	Description
DNAME	23	16 DUP(?)	DB	ASCII name for destination
PNCID	33	?	DW	Equal to SNCID
DID	35	?	DW	Number that is incremented by 1 for each frame
SNCID	37	?	DW	Field ID for frame
DNODEID	39	?	DW	Low address
	3B	?	DW	Mid address
	3D	?	DW	High address
SNODEID	3F	?	DW	Low address
	41	?	DW	Mid address
	43	?	DW	High address
PNODEID	45	?	DW	Low address
	47	?	DW	Mid address
	49	?	DW	High address
CRC	4B	?	DD	Check byte
EDF	4F	Hex 7E	DB	End-of-frame byte

Figure 3-5 (Part 2 of 2). Name Claim and Name Claim Cancel Frame

Nonexclusive Name Claim and Nonexclusive Name Claim Cancel Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 50XX	DW	Fixed value for this position of frame
N / A	11	?	DB	Hex 10 = Nonexclusive Name Claim frame Hex A0 = Nonexclusive Name Claim Cancel frame
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept
CONNID	13	?	DW	Field ID for frame. Choose a new CONNID value on the first transmission or repeat the previous CONNID on retransmissions.
N / A	15	Hex 0202	DW	Fixed value for this position of frame
N / A	17	Hex XXXX	DW	Don't-care value for this position of frame
N / A	19	Hex 0600	DW	Fixed value for this position of frame
N / A	1B	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1D	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1F	Hex 10XX	DW	Fixed value for this position of frame
N / A	21	Hex 0000	DW	Fixed value for this position of frame

Figure 3-6 (Part 1 of 2). Nonexclusive Name Claim and Nonexclusive Name Claim Cancel Frame

Field Name	Offset (Hex)	Length	Type	Description
DNAME	23	16 DUP(?)	DB	ASCII name for destination
PNCID	33	?	DW	Equal to SNCID
DID	35	?	DW	Number that is incremented by 1 for each frame
SNCID	37	?	DW	Field ID for frame
DNODEID	39	?	DW	Low address
	3B	?	DW	Mid address
	3D	?	DW	High address
SNODEID	3F	?	DW	Low address
	41	?	DW	Mid address
	43	?	DW	High address
PNODEID	45	?	DW	Low address
	47	?	DW	Mid address
	49	?	DW	High address
CRC	4B	?	DD	Check byte
EDF	4F	Hex 7E	DB	End-of-frame byte

Figure 3-6 (Part 2 of 2). Nonexclusive Name Claim and Nonexclusive Name Claim Cancel Frame

Positive Name Query Response Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01	?	DW	Low address
	03	?	DW	Mid address
	05	?	DW	High address
SADDR	07	?	DW	Low address
	09	?	DW	Mid address
	0B	?	DW	High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 30XX	DW	Fixed value for this position of frame
N / A	11	Hex 20	DB	Fixed value for this position of frame
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
SCONNID	15	?	DW	Connection ID of session
N / A	17	Hex XXXX	DW	Don't-care value for this position of frame
N / A	19	Hex 0101	DW	Fixed value for this position of frame
N / A	1B	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1D	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1F	Hex 10XX	DW	Fixed value for this position of frame
N / A	21	Hex 0010	DW	Fixed value for this position of frame
DNAME	23	16 DUP(?)	DB	ASCII name for destination
SNAME	33	16 DUP(?)	DB	ASCII name for source
DNCID	43	?	DW	Number used to determine the session for frame or to identify the datagram

Figure 3-7 (Part 1 of 2). Positive Name Query Response Frame

Field Name	Offset (Hex)	Length	Type	Description
SNCID	45	?	DW	Field ID for frame
SNODEID	47	?	DW	Low address
	49	?	DW	Mid address
	4B	?	DW	High address
CRC	4D	?	DD	Check byte
EDF	51	Hex 7E	DB	End-of-frame byte

Figure 3-7 (Part 2 of 2). Positive Name Query Response Frame

Negative Name Query Response Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01	?	DW	Low address
	03	?	DW	Mid address
	05	?	DW	High address
SADDR	07	?	DW	Low address
	09	?	DW	Mid address
	0B	?	DW	High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 40XX	DW	Fixed value for this position of frame
N / A	11	Hex 30	DB	Fixed value for this position of frame
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
N / A	15	Hex XXXX	DW	Don't care value for this position of frame
NACKREAS	17	?	DB	Reason why frame was not acknowledged
N / A	18	Hex XX	DB	Don't-care value for this position of frame
N / A	19	Hex 0101	DW	Fixed value for this position of frame
N / A	1B	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1D	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1F	Hex 10XX	DW	Fixed value for this position of frame
N / A	21	Hex XX10	DW	Fixed value for this position of frame
DNAME	23	16 DUP(?)	DB	ASCII name for destination

Figure 3-8 (Part 1 of 2). Negative Name Query Response Frame

Field Name	Offset (Hex)	Length	Type	Description
SNAME	33	16 DUP(?)	DB	ASCII name for source
DNCID	43	?	DW	Number to determine the session for the frame or to identify the datagram
CRC	45	?	DD	Check byte
EDF	47	Hex 7E	DB	End-of-frame byte

Figure 3-8 (Part 2 of 2). Negative Name Query Response Frame

Name Claim Response Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address This is the next node address (initiator's address)
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 40XX	DW	Fixed value for this position of frame
N / A	11	Hex 30	DB	Fixed value for this position of frame
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
N / A	15	Hex XXXX	DW	Don't care value for this position of frame
NACKREAS	17	?	DB	Reason why frame was not acknowledged
N / A	18	Hex XX	DB	Don't-care value for this position of frame
N / A	19	Hex 0401	DW	Fixed value for this position of frame
N / A	1B	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1D	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1F	Hex 10XX	DW	Fixed value for this position of frame
N / A	21	Hex 0000	DW	Fixed value for this position of frame

Figure 3-9 (Part 1 of 2). Name Claim Response Frame

Field Name	Offset (Hex)	Length	Type	Description
DNAME	23	16 DUP(?)	DB	ASCII name for destination
DNCID	33	?	DW	Number used to determine the session for the frame or to identify the datagram
CRC	35	?	DD	Check byte
EDF	39	Hex 7E	DB	End-of-frame byte

Figure 3-9 (Part 2 of 2). Name Claim Response Frame

Name Query Cancel Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address This is the next node address (initiator's address)
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 50XX	DW	Fixed value for this position of frame
N / A	11	Hex A0	DB	Fixed value for this position of frame
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept.
CONNID	13	?	DW	Field ID for frame. Choose a new CONNID value on the first transmission or repeat the previous CONNID on retransmissions.
N / A	15	Hex 0202	DW	Fixed value for this position of frame
N / A	17	Hex XXXX	DW	Don't-care value for this position of frame
N / A	19	Hex 0100	DW	Fixed value for this position of frame
N / A	1B	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1D	Hex XXXX	DW	Don't-care value for this position of frame
N / A	1F	Hex 10XX	DW	Fixed value for this position of frame
N / A	21	Hex XX10	DW	Fixed value for this position of frame

Figure 3-10 (Part 1 of 2). Name Query Cancel Frame

Field Name	Offset (Hex)	Length	Type	Description
DNAME	23	16 DUP(?)	DB	ASCII name for destination
SNAME	33	16 DUP(?)	DB	ASCII name for source
PNCID	43	?	DW	See definition
DID	45	?	DW	Number that is incremented by 1 for each frame
SNCID	47	?	DW	Field ID for frame
DNODEID	49	?	DW	Low address
	4B	?	DW	Mid address
	4D	?	DW	High address
SNODEID	4F	?	DW	Low address
	51	?	DW	Mid address
	53	?	DW	High address
PNODEID	55	?	DW	Low address
	57	?	DW	Mid address
	59	?	DW	High address
CRC	5B	?	DD	Check byte
EDF	5F	Hex 7E	DB	End-of-frame byte

Figure 3-10 (Part 2 of 2). Name Query Cancel Frame

Session Request Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address This is the next node address (initiator's address)
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 40XX	DW	Fixed value for this position of frame
POLL	11	Hex 0?	DB	Hex 00 to hex 07 means no poll. Hex 08 to hex 0F means to send a return frame (poll).
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
SEQ	15	?	DB	Session frame sequence number
ACK	16	?	DB	Number that includes number + 1 modulo 256 of last correctly received frame
N / A	17	Hex 0001	DW	Fixed value for this position of frame
PKTSIZE	19	?	DW	Frame size that can be accepted from the remote node
N / A	1B	Hex 0000	DW	Fixed value for this position of frame
N / A	1D	Hex 1010	DW	Fixed value for this position of frame
SNAME	1F	16 DUP(?)	DB	ASCII name for source

Figure 3-11 (Part 1 of 2). Session Request Frame

Field Name	Offset (Hex)	Length	Type	Description
DNAME	2F	16 DUP(?)	DB	ASCII name for destination
DNCID	3F	?	DW	Number used to determine the session for the frame or to identify the datagram
CRC	41	?	DD	Check byte
EDF	45	Hex 7E	DB	End-of-frame byte

Figure 3-11 (Part 2 of 2). Session Request Frame

Session Accept Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address This is the next node address (initiator's address)
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 40XX	DW	Fixed value for this position of frame
POLL	11	Hex 0?	DB	Hex 00 to hex 07 means no poll. Hex 08 to hex 0F means to send a return frame (poll).
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	DUP (0)	DW	Field ID for frame
SEQ	15	?	DB	Session frame sequence number
ACK	16	?	DB	Number that includes number + 1 modulo 256 of last correctly received frame
N / A	17	Hex 0002	DW	Fixed value for this position of frame
PKTSIZE	19	?	DW	Frame size that can be accepted from the remote node
DNCID	1B	?	DW	Number used to determine the session for the frame or to identify the datagram
CRC	1D	?	DD	Check byte
EDF	22	Hex 7E	DB	End-of-frame byte

Figure 3-12. Session Accept Frame

Session Reject Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01	?	DW	Low address
	03	?	DW	Mid address
	05	?	DW	High address
SADDR	07	?	DW	Low address
	09	?	DW	Mid address
	0B	?	DW	High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 40XX	DW	Fixed value for this position of frame
POLL	11	Hex 0?	DB	Hex 00 to hex 07 means no poll. Hex 08 to hex 0F means to send a return frame (poll).
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
SEQ	15	?	DB	Session frame sequence number
ACK	16	?	DB	Number that includes number + 1 modulo 256 of last correctly received frame
N / A	17	Hex 03	DB	Fixed value for this position of frame
RVAL	18	?	DB	Indication of why the session was rejected
DNCID	19	?	DW	Number used to determine the session for the frame or to identify the datagram
CRC	1B	?	DD	Check byte
EDF	1F	Hex 7E	DB	End-of-frame byte

Figure 3-13. Session Reject Frame

ACK Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address This is the next node address (initiator's address)
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 4000	DW	Fixed value for this position of frame
POLL	11	Hex 4?	DB	Hex 40 to hex 47 means no poll. Hex 48 to hex 4F means to send a return frame (poll).
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
SEQ	15	?	DB	Session frame sequence number
ACK	16	?	DB	Number that includes number +1 modulo 256 of last correctly received frame
N / A	17	Hex XX	DB	Don't care value for this position of frame
DNCID	18	?	DW	Number used to determine the session for the frame or to identify the datagram
CRC	1A	?	DD	Check byte
EDF	1E	Hex 7E	DB	End-of-frame byte

Figure 3-14. ACK Frame

NACK Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01	?	DW	Low address
	03	?	DW	Mid address
	05	?	DW	High address
SADDR	07	?	DW	Low address
	09	?	DW	Mid address
	0B	?	DW	High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 4000	DW	Fixed value for this position of frame
POLL	11	Hex 5?	DB	Hex 50 to hex 57 means no poll. Hex 58 to hex 5F means to send a return frame (poll).
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
SEQ	15	?	DB	Session frame sequence number
ACK	16	?	DB	Number that includes number +1 modulo 256 of last correctly received frame
NACKREAS	17	?	DB	Reason why the frame was not received
DNCID	18	?	DW	Number used to determine the session for the frame or to identify the datagram
CRC	1A	?	DD	Check byte
EDF	1E	Hex 7E	DB	End-of-frame byte

Figure 3-15. NACK Frame

Close Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address This is the next node address (initiator's address)
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 4000	DW	Fixed value for this position of frame
POLL	11	Hex 6?	DB	Hex 60 to hex 67 means no poll. Hex 68 to hex 6F means to send a return frame (poll).
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
SEQ	15	?	DB	Session frame sequence number
ACK	16	?	DB	Number that includes number +1 modulo 256 of last correctly received frame
N / A	17	Hex XX	DB	Fixed value for this position of the frame
DNCID	18	?	DW	Number used to determine the session for the frame or to identify the datagram
CRC	1A	?	DD	Check byte
EDF	1E	Hex 7E	DB	End-of-frame byte

Figure 3-16. Close Frame

Closed Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address This is the next node address (initiator's address)
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 4000	DW	Fixed value for this position of frame
N / A	11	Hex 70	DB	Fixed value for this position of frame
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
SEQ	15	?	DB	Session frame sequence number
ACK	16	?	DB	Number that includes number + 1 modulo 256 of last correctly received frame
CLREAS	17	?	DB	Reason why connection closed
DNCID	18	?	DW	Number used to determine the session for the frame or to identify the datagram
CRC	1A	?	DD	Check byte
EDF	1E	Hex 7E	DB	End-of-frame byte

Figure 3-17. Closed Frame

Data Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address This is the next node address (initiator's address)
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 40XX	DW	Fixed value for this position of frame
POLL	11	Hex 0?	DB	Hex 00 to hex 07 means no poll. Hex 08 to hex 0F means to send a return frame (poll).
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
SEQ	15	?	DB	Session frame sequence number
ACK	16	?	DB	Number that includes number + 1 modulo 256 of last correctly received frame
EOM	17	Hex ?0	DB	End-of-message field. Hex (80-F0) equals end-of-message.
DATA	18	?? DUP(?)	DB	Variable length field
DNCID	XX	?	DW	Number used to determine the session for the frame or to identify the datagram
CRC	XX	?	DD	Check byte
EDF	XX	Hex 7E	DB	End-of-frame byte

Figure 3-18. Data Frame

Datagram Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01	?	DW	Low address
	03	?	DW	Mid address
	05	?	DW	High address
SADDR	07	?	DW	Low address
	09	?	DW	Mid address
	0B	?	DW	High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 51XX	DW	Fixed value for this position of frame
N / A	11	Hex 0100	DW	Fixed value for this position of frame
N / A	13	Hex 0001	DW	Fixed value for this position of frame
N / A	15	Hex 1010	DW	Fixed value for this position of frame
N / A	17	Hex 0000	DW	Fixed value for this position of frame
SNAME	19	16 DUP(?)	DB	ASCII name for source
DNAME	29	16 DUP(?)	DB	ASCII name for destination
DATA	39	?? DUP(?)	DB	Variable length field
PNCID	XX	?	DW	Equal to SNCID
DID	XX	?	DW	Number that is incremented by 1 for each frame
SNCID	XX	Hex FFFE	DW	Field ID for frame
DNODEID	XX	?	DW	Low address
	XX	?	DW	Mid address
	XX	?	DW	High address
SNODEID	XX	?	DW	Low address
	XX	?	DW	Mid address
	XX	?	DW	High address
PNODEID	XX	?	DW	Low address
	XX	?	DW	Mid address
	XX	?	DW	High address

Figure 3-19 (Part 1 of 2). Datagram Frame

Field Name	Offset (Hex)	Length	Type	Description
CRC	XX	?	DD	Check byte
EDF	XX	Hex 7E	DB	End-of-frame byte

Figure 3-19 (Part 2 of 2). Datagram Frame

Get Status Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 51XX	DW	Fixed value for this position of frame
N / A	11	Hex 0300	DW	Fixed value for this position of frame
N / A	13	Hex 0003	DW	Fixed value for this position of frame
CMDRESP	15	Hex ?2	DB	Hex (02-72) = command Hex (82-F2) = response
N / A	16	Hex XX	DB	Fixed value for this position of frame
N / A	17	Hex 8001	DW	Fixed value for this position of frame
N / A	19	Hex 8001	DW	Fixed value for this position of frame
TRANSID	1B	?	DW	ID for status and status response frame
N / A	1D	Hex 10	DB	Fixed value for this position of frame
DNAME	1E	16 DUP(?)	DB	ASCII name for destination
N / A	2E	Hex 00	DB	Fixed value for this position of frame
PNCID	2F	Hex FFFE	DW	Equal to SNCID
DID	31	?	DW	Number that is incremented by 1 for each frame
SNCID	33	Hex FFFE	DW	Field ID for frame

Figure 3-20 (Part 1 of 2). Get Status Frame

Field Name	Offset (Hex)	Length	Type	Description
DNODEID	35	?	DW	Low address
	37	?	DW	Mid address
	39	?	DW	High address
SNODEID	3B	?	DW	Low address
	3D	?	DW	Mid address
	3F	?	DW	High address
PNODEID	41	?	DW	Low address
	43	?	DW	Mid address
	45	?	DW	High address
CRC	47	?	DD	Check byte
EDF	4B	Hex 7E	DB	End-of-frame byte

Figure 3-20 (Part 2 of 2). Get Status Frame

Status Response Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01	?	DW	Low address
	03	?	DW	Mid address
	05	?	DW	High address
SADDR	07	?	DW	Low address
	09	?	DW	Mid address
	0B	?	DW	High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 51XX	DW	Fixed value for this position of frame
N / A	11	Hex 0300	DW	Fixed value for this position of frame
N / A	13	Hex 0003	DW	Fixed value for this position of frame
N / A	15	Hex 82	DB	Fixed value for this position of frame
N / A	16	Hex 00	DB	Fixed value for this position of frame
N / A	17	Hex 8001	DW	Fixed value for this position of frame
N / A	19	Hex 8001	DW	Fixed value for this position of frame
TRANSID	1B	?	DW	The transaction identification number
N / A	1D	Hex 10	DB	Fixed value for this position of frame
DNAME	1E	16 DUP(?)	DB	ASCII name for destination
N / A	2E	Hex 00	DB	Fixed value for this position of frame
STATLEN	2F	?	DW	Equal to the offset value of NRALIAS minus hex 12
DNODEID	31	?	DW	Low address
	33	?	DW	Mid address
	35	?	DW	High address

Figure 3-21 (Part 1 of 4). Status Response Frame

Field Name	Offset (Hex)	Length	Type	Description
JUMPSTAT	37	?	DB	Two high-order bits, indicating the position of jumpers W1 and W2. When set to 1, the jumpers are in place.
SELFTEST	38	?	DB	The result of the node's self-test
SWVERSION	39	?	DW	The software version used by the node
REPORTPD	3B	?	DW	The time in minutes since the last hardware reset
CRCERRS	3D	?	DW	Number of properly aligned frames received with a CRC error
ALGNERRS	3F	?	DW	Number of misaligned frames received with a CRC error
COLLISIONS	41	?	DW	Number of collisions encountered on transmission frames
XMITABRTS	43	Hex 0000	DW	Number of transmitted frames that transmitted unsuccessfully
XMITMSGS	45	?	DD	Number of frames successfully transmitted by the Link level
RECVMSGS	49	?	DD	Number of successfully received frames
REXMITCNT	4D	?	DW	Number of frames retransmitted
NORESOURCES	4F	?	DW	The number of frames discarded because of a lack of memory resources
N / A	51	Hex XXXX	DW	Don't-care value for this position of frame
N / A	53	Hex XXXX	DW	Don't-care value for this position of frame
N / A	55	Hex XXXX	DW	Don't-care value for this position of frame
N / A	57	Hex XXXX	DW	Don't-care value for this position of frame

Figure 3-21 (Part 2 of 4). Status Response Frame

Field Name	Offset (Hex)	Length	Type	Description
N / A	59	Hex XXXX	DW	Don't-care value for this position of frame
N / A	5B	Hex XXXX	DW	Don't-care value for this position of frame
N / A	5D	Hex XXXX	DW	Don't-care value for this position of frame
N / A	5F	Hex XXXX	DW	Don't-care value for this position of frame
N / A	61	Hex XXXX	DW	Don't-care value for this position of frame
N / A	63	Hex XXXX	DW	Don't-care value for this position of frame
N / A	65	Hex XXXX	DW	Don't-care value for this position of frame
N / A	67	Hex XXXX	DW	Don't-care value for this position of frame
N / A	69	Hex XXXX	DW	Don't-care value for this position of frame
NRALIAS	6B	?	DB	Number of alias names to follow in the frame. (The next three fields, ALIASNAME, ALIASNR, and ALIASTAT, are repeated in sequence with the number given in the NRALIAS field. This field's offset can be calculated by using the STATLEN field offset and adding hex 12.
ALIASNAME	6C	16 DUP(?)	DB	Alias name for 16 bytes
ALIASNR	7C	?	DB	Number assigned to the alias name
ALIASTAT	7D	?	DB	The status of the name specified in the ALIASNAME field
O	XX	O	O	
O	XX	O	O	
O	XX	O	O	

Figure 3-21 (Part 3 of 4). Status Response Frame

Field Name	Offset (Hex)	Length	Type	Description
ALIASNAME	XX	16 DUP(?)	DB	Alias name for 16 bytes
ALIASNR	XX	?	DB	Number assigned to the alias name
ALIASTAT	XX	?	DB	The status of the name specified in the ALIASNAME field
PNCID	XX	Hex FFFE	DW	See definition
DID	XX	?	DW	Number that is incremented by 1 for each frame
SNCID	XX	Hex FFFE	DW	Field ID for frame
DNODEID	XX	?	DW	Low address
	XX	?	DW	Mid address
	XX	?	DW	High address
SNODEID	XX	?	DW	Low address
	XX	?	DW	Mid address
	XX	?	DW	High address
PNODEID	XX	?	DW	Low address
	XX	?	DW	Mid address
	XX	?	DW	High address
CRC	XX	?	DD	Check byte
EDF	5F	Hex 7E	DB	End-of-frame byte

Figure 3-21 (Part 4 of 4). Status Response Frame

Abort Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address This is the next node address (initiator's address).
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 40XX	DW	Fixed value for this position of frame
POLL	11	Hex 9?	DB	Hex (90-97) means no poll. Hex (98-9F) means to send a return frame.
WIN	12	Hex 0?	DB	Low-order 4 bits define the number of frames the sender will accept. High-order 4 bits are a fixed value.
CONNID	13	?	DW	Field ID for frame
SEQ	15	?	DB	Session frame sequence number
ACK	16	?	DB	Number that includes number + 1 modulo 256 of last correctly received frame
N / A	17	Hex XX	DB	Don't-care value for this position of frame
DNCID	18	?	DW	Number used to determine the session for the frame or to identify the datagram
CRC	1A	?	DD	Check byte
EDF	1E	Hex 7E	DB	End-of-frame byte

Figure 3-22. Abort Frame

Self-Test Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01	?	DW	Low address
	03	?	DW	Mid address
	05	?	DW	High address
SADDR	07	?	DW	Low address
	09	?	DW	Mid address
	0B	?	DW	High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex F0XX	DW	Fixed value for this position of frame
CRC	11	?	DD	Check byte
EDF	15	Hex 7E	DB	End-of-frame byte

Figure 3-23. Self-Test Frame

Ident Frame

Field Name	Offset (Hex)	Length	Type	Description
STD	00	Hex 7E	DB	Start delimiter flag byte
DADDR	01 03 05	? ? ?	DW DW DW	Low address Mid address High address
SADDR	07 09 0B	? ? ?	DW DW DW	Low address Mid address High address
DLEN	0D	?	DW	Length of Link level frame
N / A	0F	Hex 51XX	DW	Fixed value for this position of frame
N / A	11	Hex 0300	DW	Fixed value for this position of frame
N / A	13	Hex 0003	DW	Fixed value for this position of frame
CMDRESP	15	Hex ?1	DB	Hex 41 = command Hex C1 = response
N / A	16	Hex 00	DB	Fixed value for this position of frame
N / A	17	Hex 8001	DW	Fixed value for this position of frame
N / A	19	Hex 8001	DW	Fixed value for this position of frame
TRANSID	1B	Hex 0000	DW	ID for status and status response frame
N / A	1D	Hex 10	DB	Fixed value for this position of frame
DNAME	1E	16 DUP(?)	DB	ASCII name for destination
N / A	2E	Hex 00	DB	Fixed value for this position of frame
NODEIDMASK	2F 31 33	? ? ?	DW DW DW	Mask value Mask value Mask value
SNAME	33	16 DUP(?)	DB	ASCII name for source

Figure 3-24 (Part 1 of 2). Ident Frame

Field Name	Offset (Hex)	Length	Type	Description
NODEIDMATCH	35	?	DW	Match value
	37	?	DW	
	39	?	DW	
PNCID	3B	Hex FFFE	DW	Equal to SNCID
DID	3D	?	DW	Number that is incremented by 1 for each frame
SNCID	3F	Hex FFFE	DW	Field ID for frame
DNODEID	41	?	DW	Low address
	43	?	DW	Mid address
	45	?	DW	High address
SNODEID	47	?	DW	Low address
	49	?	DW	Mid address
	4B	?	DW	High address
PNODEID	4D	?	DW	Low address
	4F	?	DW	Mid address
	51	?	DW	High address
CRC	53	?	DD	Check byte
EDF	57	Hex 7E	DB	End-of-frame byte

Figure 3-24 (Part 2 of 2). Ident Frame

Glossary of Terms and Abbreviations

@. Address

address. A character or a group of characters that identifies a register, a particular part of storage, or some other data source or destination. (2) To refer to a device or an item of data by its address. (3) The part of the selection signals that indicates the destination of a call. (4) In word processing, the location, identified by the address code, of a specific section of the recording medium or storage.

alias. An alternate name that you can be known by on the network; synonymous with name.

alignment error. The number of misaligned frames received with a CRC error. When excessive or missing bits occur during the reception of a frame, the frame is misaligned.

application program. (1) A program written for or by a user that applies to the user's work. (2) A program used to connect and communicate with stations in a network, enabling users to perform application-oriented activities.

BIOS. Basic Input Output System

buffer. (1) An area of storage that is temporarily reserved for use in performing an input/output operation, into which data is read or from which data is written. Synonymous with I/O area. (2) A portion of storage for temporarily holding input or output data.

collisions. When a frame from a transmitting adapter encounters any other signal in its path (frame, noise, or another type of signal), the adapter stops transmitting and a collision is registered.

command. (1) A control signal. (2) A request from a node for the performance of an operation or the execution of a particular program.

configuration. (1) The arrangement of a computer system or network as defined by the nature, number and the chief characteristics of its functional units. More specifically, the term configuration may refer to a hardware configuration or a software configuration. (2) The devices and programs that make up a system, subsystem, or network.

CRC. Cyclic Redundancy Check

CRC errors. The number of properly aligned frames received with a cyclic redundancy check.

cyclic redundancy check (CRC). (1) A redundancy check in which the check key is generated by a cyclic algorithm. (2) A system of error checking performed at both the sending and receiving station after a block-check character has been accumulated.

datagram. A particular type of information encapsulation at the network layer of the adapter protocol for NETBIOS. No explicit acknowledgment for the information is sent by the receiver. Instead, transmission relies on the "best effort" of the link layer.

data transfer. (1) The result of the transmission of data signals from any data source to a data sink. (2) The movement, or copying, of data from one location and the storage of the data at another location.

DB. Define Byte

DD. Define Double-Word

default. An alternative value, attribute, or option that is assumed when none has been specified.

default value. The choice among exclusive alternatives made by the system when no explicit choice is specified by the user.

define byte (DB). When defining 8086 data type control blocks this assembler pseudo-operator defines a single byte and byte strings.

define double-word (DD). When defining 8086 data type control blocks this assembler pseudo-operator defines personal computer storage addresses.

define word (DW). When defining 8086 data type control blocks this assembler pseudo-operator defines numeric fields that require two bytes.

device. An input/output unit such as a terminal, display, or printer.

duplex. (1) In data communication, pertaining to a simultaneous two-way independent transmission in both directions. (2) Contrast with half-duplex.

DW. Define Word

exhausted resources. The number of frames discarded because of a lack of memory.

field. (1) In a record, a specified area used for a particular category of data. (2) In a data base, the smallest unit of data that can be referred to.

frame. (1) In SDLC, the vehicle for every command, every response, and all information that is transmitted using SDLC procedures. Each frame begins and ends with a flag. (2) In data transmission, the sequence of contiguous bits bracketed by and including beginning and ending flag sequences.

function. (1) A mathematical entity whose dependent variable value depends in a specified manner on the values of one or more independent variables, not more than one value of the dependent variable corresponding to each permissible combination of values from the respective ranges of the independent variables. (2) A specific purpose of an entity, or its characteristic action. (3) In data communications, a machine action such as carriage return or line feed. (4) In computer programming, synonym for procedure.

hot carrier. The condition resulting from a transmitter being locked in transmit mode.

interaction. A basic unit used to record system activity, consisting of the acceptance of a line of terminal input, processing of the line, and a response, if any.

interrupt. (1) A suspension of a process, such as the execution of a computer program, caused by an event external to that process, and performed in such a way that the process can be resumed. (2) In a data transmission, to take an action at a receiving station that causes the transmitting station to terminate a transmission. (3) Synonymous with interruption.

layer. (1) In open systems architecture, a collection of related functions that comprise one level of a hierarchy of functions. Each layer specifies its own functions and assumes that lower level functions are provided. (2) A grouping of related functions that are logically separate from the functions of other layers; the implementation of the functions in one layer can be changed without affecting functions in other layers.

local area network. A communication system that links devices and allows them to exchange information and share resources, such as a fixed disk. Devices are connected by a cable through which information is sent and received.

local session number. The number assigned to each session established by an adapter. Each session receives a unique number that distinguishes it from any other active sessions.

message. A message is a logical partition of the user device's data stream to and from the adapter.

multitasking. (1) Pertaining to the concurrent execution of two or more tasks by a computer. (2) Multiprogramming that provides for the concurrent performance, or interleaved execution, of two or more tasks.

NCB. Network Control Block

NETBIOS. Network Basic Input Output System

network. An interconnected group of nodes.

node. (1) In a network, a point where one or more functional units interconnect transmission lines. (2) Any device attached to a network that transmits or receives data.

operating system. Software that controls the execution of programs; an operating system may provide services such as resource allocation, scheduling, input/output control, and data management.

option. A specification in a statement that may be used to influence the execution of the statement.

parameter. A variable that is given a constant value for a specified application and that may denote the application.

point-to-point. A connection between two nodes on a network.

POST. Power-On Self-Test

post. To affix to a usual place. Used with NETBIOS to indicate providing items such as return code at the end of a command or function.

protocol. (1) A specification for the format and relative timing of information exchanged between communicating parties. (2) The set of rules governing the operation of functional units of a communication system that must be followed if communication is to be achieved.

pseudocode. A code that requires translation prior to execution.

RAM. Random Access Memory

register. (1) A storage device, having a specified storage capacity such as a bit, a byte, or a computer word, and usually intended for a special purpose. (2) A storage device in which specific data is stored.

reserved names. A name that cannot be added or deleted. For example, any name starting with an asterisk (*) or binary 00.

return code. A code used to influence the execution of succeeding instructions.

routine. (1) Part of a program, or a sequence of instructions called by a program, that may have some general or frequent use. (2) The forwarding of a message unit along a particular path through a network, as determined by the parameters carried in the message unit, such as the destination network address in a transmission header.

ROM. Read-Only Memory

RPL. Remote Program Load

session. (1) A connection between two nodes that allows them to communicate. (2) The period of time during which a user of a node can communicate with an interactive system; usually, the elapsed time between logon and logoff. (3) A logical connection between two network addressable units that can be activated, tailored to provide various protocols, and deactivated as requested.

successful receptions. The number of frames successfully received.

successful transmissions. The number of frames successfully transmitted.

throughput. A measure of the amount of work performed by a computer system over a given period of time, for example, jobs per day.

time-out. (1) A parameter related to an enforced event designed to occur at the conclusion of a predetermined elapsed time. A time-out condition can be canceled by the receipt of an appropriate time-out cancelation signal. (2) A time interval allotted for certain operations to occur; for example, response to polling or addressing before system operation is interrupted and must be restarted.

variable. (1) In computer programming, a character or group of characters that refers to a value and, in the execution of a computer

program, corresponds to an address. (2) A quantity that can assume any of a given set of values.

virtual connection. A connection between two nodes on the network that is established using the transport layer and provides reliable data between nodes.

Index

O

P

R

S

T

Printed in the
United States of America

References in this publication to IBM products or services do not imply that IBM intends to make them available outside the United States.

68X2270
S68X-2270-00