


Burroughs 

**B 1800/B 1700
Systems
Network Definition
Language (NDL)**

REFERENCE MANUAL

RELATIVE TO MARK VII.0 RELEASE

PRICED ITEM

**B 1800/B 1700
Systems
Network Definition
Language (NDL)**

REFERENCE MANUAL

RELATIVE TO MARK VII.0 RELEASE

Copyright © 1978 Burroughs Corporation, Detroit, Michigan 48232

PRICED ITEM

LIST OF EFFECTIVE PAGES

Item	Page	Item	Page
Title	Original	7-8	Blank
ii thru viii	Original	A-1	Original
1-1	Original	A-2	Blank
1-2	Blank	B-1 thru B-20	Original
2-1 thru 2-6	Original	C-1	Original
3-1 thru 3-72	Original	C-2	Blank
4-1 thru 4-34	Original	D-1 thru D-13	Original
5-1 thru 5-6	Original	D-14	Blank
6-1 thru 6-2	Original	E-1 thru E-19	Original
7-1 thru 7-7	Original	E-20	Blank
		Index-1 thru Index-4	Original

Burroughs believes that the software described in this manual is accurate and reliable, and much care has been taken in its preparation. However, no responsibility, financial or otherwise, can be accepted for any consequences arising out of the use of this material, including loss of profit, indirect, special, or consequential damages. There are no warranties which extend beyond the program specification.

The Customer should exercise care to assure that use of the software will be in full compliance with laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change. Revisions may be issued from time to time to advise of changes and/or additions.

TABLE OF CONTENTS

SECTION	TITLE	PAGE
1	INTRODUCTION	1-1
	General	1-1
	Related Document	1-1
2	GENERAL NDL INFORMATION.....	2-1
	Description of the Network Definition Language	2-1
	The Network Controller.....	2-1
	Message Control System.....	2-3
	Types of Definitions	2-3
	NDL Source Program Organization.....	2-3
	Declaration Section	2-3
	Request Section	2-3
	Line Control Section	2-4
	Terminal Section	2-4
	Station Section	2-4
	Line Section.....	2-4
	File Section	2-4
	Statement Requirements and Defaults	2-5
3	STRUCTURE OF THE LANGUAGE.....	3-1
	Syntax Conventions.....	3-1
	Metalinguistic Symbols	3-1
	Basic Elements.....	3-1
	Character Set	3-1
	Basic Components	3-2
	Identifiers	3-2
	Integers	3-3
	Strings	3-3
	Operators.....	3-3
	Logical Values	3-4
	Brackets.....	3-4
	Separators	3-5
	Variables	3-5
	System Status Variables.....	3-6
	CHAR, CHARACTER	3-7
	FREQUENCY (INPUT), FREQUENCY (OUTPUT)	3-7
	IODESC	3-8
	INPUTATTACHED	3-8
	OUTPUTATTACHED	3-8
	LENGTH (INPUT), LENGTH (OUTPUT)	3-8
	LINE	3-8
	LINE (CONTROL KEY)	3-8
	LINE (QUEUED)	3-10
	MAXSTATIONS	3-10
	RESULTDESC	3-10
	RETRY	3-10
	SEQUENCE	3-10
	STATION	3-10
	STATION (ENABLED)	3-11
	STATION (LINE)	3-11
	STATION (MYUSE)	3-11
	STATION (QUEUED)	3-11
	STATION (READY)	3-11
	STATION (TYPE)	3-12
	STATION (VALID)	3-13
	TERMINALTYPE	3-13
	TIME	3-13

TABLE OF CONTENTS (Cont)

SECTION	TITLE	PAGE
3 (Cont)	TRAN (RECEIVE)	3-13
	TRAN (TRANSMIT)	3-13
	Error Flags	3-14
	ADDERR	3-14
	BREAK	3-14
	ACCESSERR	3-14
	ENDOFBUFFER	3-14
	EXCEPTION	3-15
	FORMATERR	3-15
	LOSSOFCARRIER	3-15
	LOSSOFDSR	3-15
	PARITY	3-15
	TIMEOUT	3-15
	TRANERR	3-15
	User Toggles	3-15
	TOG [< TOG INDEX >]	3-15
	LINE(TOG [< TOG INDEX >])	3-15
	User Tallies	3-16
	TALLY [< TALLY INDEX >]	3-16
	LINE(TALLY [< TALLY INDEX >])	3-16
	TIME(TALLY)	3-16
	String Functions	3-17
	DECIMAL	3-17
	CONVERT	3-18
	Expressions	3-19
	Syntax and Semantics of Sections	3-20
	Declaration Section	3-21
	AUDITFILE Declaration	3-21
	CONSTANT Declaration	3-23
	DEFINE Statement	3-25
	MAX TALLY/TOG Declaration	3-26
	MAX BUFFERS Declaration	3-27
	MAX FILES Declaration	3-27
	MAX MESSAGES Declaration	3-27
	NIF NAME Declaration	3-28
	Request Section	3-29
	ASSIGNMENT Statement	3-30
	ATTACH OUTPUT Statement	3-30
	AUDIT Statement	3-30
	CASE Statement	3-32
	DECREMENT Statement	3-33
	DETACH OUTPUT Statement	3-33
	DISPLAY Statement	3-33
	DO Statement	3-34
	DUMP Statement	3-35
	FETCH Statement	3-35
	FINISH Statement	3-36
	IF Statement	3-36
	INCREMENT Statement	3-37
	INITIALIZE Statement	3-37
	INITIATE Statement	3-38
	NULL Statement	3-39
	QUEUE INPUT Statement	3-39
	QUEUE OUTPUT Statement	3-39

TABLE OF CONTENTS (Cont)

SECTION	TITLE	PAGE
3 (Cont)	RECEIVE Statement	3-40
	TERMINATE Statement	3-41
	TRANSMIT Statement	3-43
	UNDO Statement	3-45
	Line Control Section	3-48
	ASSIGNMENT Statement	3-49
	CASE Statement	3-49
	CONTINUE Statement	3-49
	DISPLAY Statement	3-50
	DO Statement	3-50
	DUMP Statement	3-51
	IF Statement	3-51
	INITIATE Statement	3-52
	NULL Statement	3-54
	POLL Statement	3-54
	UNDO Statement	3-56
	Terminal Section	3-57
	BUFFERSIZE Statement	3-58
	TERMINAL ADDRESS Statement	3-58
	TERMINAL DEFAULT Statement	3-58
	TERMINAL DIAGNOSTIC REQUEST Statement	3-59
	TERMINAL REQUEST Statement	3-60
	TERMINAL TYPE Statement	3-61
	TRANSMISSION NUMBER Statement	3-61
	Station Section	3-62
	CONTROLLER Statement	3-62
	FREQUENCY Statement	3-63
	LOGICALACK Statement	3-64
	MYUSE Statement	3-64
	RETRY Statement	3-65
	STATION ADDRESS Statement	3-65
	STATION DEFAULT Statement	3-66
	STATION READY Statement	3-66
	STATION TERMINAL Statement	3-66
	Line Section	3-67
	AUTOPOLL SIZE Statement	3-68
	LINE ADDRESS Statement	3-68
	LINE CONTROL Statement	3-69
	LINE DEFAULT Statement	3-69
	LINE STATION Statement	3-69
	LINE TYPE Statement	3-70
	File Section	3-71
	FAMILY Statement	3-71
	FILE DEFAULT Statement	3-72
	RESIDENT Statement	3-72
4	MESSAGE CONTROL SYSTEM	4-1
	Description of Message Control Systems	4-1
	The Remote File Interface	4-1
	Sharing Resources Among Multiple MCS Files	4-2
	MCS Messages and Replies	4-2
	General	4-2
	Data Messages	4-5
	General	4-5
	Data Message Format	4-6

TABLE OF CONTENTS (Cont)

SECTION	TITLE	PAGE
4 (Cont)	Semantics of the Data Messages	4-7
	OPEN and OPEN-REPLY Messages	4-9
	General	4-9
	Format of the OPEN and OPEN-REPLY Messages	4-11
	Semantics of the OPEN and OPEN-REPLY Messages	4-12
	Network Controller OPEN Review Criteria	4-14
	ATTACH and ATTACH-REPLY Messages	4-15
	General	4-15
	Remote Files Associated with ATTACH	4-15
	Format of the ATTACH Message	4-18
	Format of the ATTACH-REPLY Message	4-19
	Semantics of the ATTACH and ATTACH-REPLY Messages	4-20
	Network Controller ATTACH Review Criteria	4-21
	DETACH and DETACH-REPLY Messages	4-22
	General	4-22
	Format of the DETACH and DETACH-REPLY Messages	4-23
	Semantics of the DETACH and DETACH-REPLY Messages	4-24
	CLOSE Message	4-24
	General	4-24
	Format of the CLOSE Message	4-25
	Semantics of the CLOSE Message	4-25
	STATUS and STATUS-REPLY Messages	4-25
	General	4-25
	Format of the STATUS and STATUS-REPLY Messages	4-26
	Semantics of the STATUS and STATUS-REPLY Messages	4-27
	CHANGE and CHANGE-REPLY Messages	4-30
	General	4-30
	Format of the CHANGE and CHANGE-REPLY Messages	4-30
	Semantics of the CHANGE and CHANGE-REPLY Messages	4-30
	RECALL, RECALL-REPLY, REMOVE, and REMOVE-REPLY Messages ..	4-31
	General	4-31
	Format of the RECALL, RECALL-REPLY, REMOVE, and REMOVE-REPLY Messages	4-32
	Semantics of the RECALL, REMOVE, RECALL-REPLY, and REMOVE-REPLY Messages	4-32
	REMOTE-FILE-INFO and REMOTE-FILE-INFO-REPLY Messages	4-32
	General	4-32
	Format of the REMOTE-FILE-INFO and REMOTE-FILE-INFO-REPLY Messages	4-33
	Semantics of the REMOTE-FILE-INFO and REMOTE-FILE-INFO-REPLY Messages	4-33
	User Messages	4-34
5	USING THE NDL COMPILER	5-1
	Compiler Input and Output Files	5-1
	Input	5-1
	Output	5-1
	NDL Control Cards	5-1
	NDL Source Card Format	5-2
	Source Code	5-2
	Comments	5-3
	Sequence Numbers	5-3
	Compiler Control Cards	5-3
	NDL Coding Aids	5-5
	Compiler-Defined Identifiers	5-5

TABLE OF CONTENTS (Cont)

SECTION	TITLE	PAGE
5 (Cont)	Default Definitions	5-6
	Library REQUEST/CONTROL Routines	5-6
6	NETWORK CONTROLLER DIAGNOSTIC AIDS	6-1
	General	6-1
	Diagnostic REQUEST Sets	6-1
	Audit Facilities	6-1
	IOLOG	6-1
	General	6-1
	Operating Instructions	6-1
7	NETWORK CONTROLLER ERROR HANDLING	7-1
	General	7-1
	Network Controller Run Errors	7-1
	NDL/DUMP Program	7-2
	General	7-2
	Data Formats	7-2
	Operating Instructions	7-3
	Dump Analysis	7-3
	Cross Reference of Variables	7-4
Appendix A –	Reserved Words	A-1
Appendix B –	Glossary	B-1
Appendix C –	List of Required Statements	C-1
Appendix D –	NDL Program Example	D-1
Appendix E –	NDL Library	E-1
	General	E-1
	Accessing the Library	E-1
	NDL/LIBRARY File	E-2
	Requests	E-2
	Input Requests	E-2
	Poll or Autopoll Request	E-2
	Dynamic Request	E-3
	Point-to-Point Requests	E-4
	Teletype Request	E-5
	Output Requests	E-5
	Select Requests	E-5
	Dynamic Request	E-6
	Fast Select Requests	E-6
	Point-to-Point Request	E-7
	Teletype Request	E-8
	Input/Output Requests	E-8
	Poll/Select Request	E-9
	Point-to-Point Request	E-9
	Teletype Request	E-10
	Remote Job Entry (RJE) Request	E-10
	Basic Line Discipline	E-10
	Error Recovery	E-11

TABLE OF CONTENTS (Cont)

SECTION	TITLE	PAGE
Appendix E (Cont)	File Control	E-11
	Variables	E-11
	RJE/HOST Request	E-11
	Controls	E-12
	Multi-Point Controls	E-12
	Poll/Select Control	E-12
	Autopoll Controls	E-13
	Dynamic Control	E-14
	Point-to-Point Controls	E-15
	Contention Control	E-15
	Conversational Control	E-16
	Remote Job Entry (RJE) Control	E-16
	Associated Requests and Controls	E-16
	Associated Terminal and Request Sets	E-18

LIST OF ILLUSTRATIONS

FIGURE	TITLE	PAGE
2-1	NDL Generation Process	2-1
2-2	The Network Controller	2-2
2-3	Detailed NDL Generation Process	2-5
3-1	Controller/Slave Stations	3-61
5-1	NDL Source Card Deck	5-3
D-1	Physical Definition of a Data Communications Network	D-1
D-2	Logical Definition of a Data Communications Network	D-2

LIST OF TABLES

TABLE	TITLE	PAGE
3-1	Characteristics of Variables	3-18
3-2	NDL Compiler – Defined Identifiers	3-24
3-3	Terminate Options	3-44
4-1	Messages The MCS Can Read	4-3
4-2	Messages The MCS Can Write	4-4
4-3	Data Message Format	4-6
4-4	OPEN And OPEN-REPLY Message Format	4-11
4-5	Station Status	4-17
4-6	Format of the ATTACH Message	4-18
4-7	Format of the ATTACH-REPLY Message	4-19
4-8	Format of the DETACH Message	4-23
4-9	Format of the CLOSE Message	4-25
4-10	Format of the STATUS and STATUS-REPLY Messages	4-26
4-11	Format of the CHANGE and CHANGE-REPLY Messages	4-30
4-12	Format of the RECALL, RECALL-REPLY, REMOVE, and REMOVE-REPLY Messages	4-32
4-13	Format of the REMOTE-FILE-INFO and REMOTE-FILE-INFO-REPLY Messages	4-33
4-14	User-Defined Message Format	4-34
7-1	Cross Reference of Variables	7-4
7-2	Correlation of NDL Attributes and NDL/DUMP Field Names	7-6
E-1	Associated REQUESTs and CONTROLS	E-17
E-2	Associated Terminal and REQUEST Sets	E-18

SECTION 1

INTRODUCTION

GENERAL

This manual provides a complete description of the B 1800/B 1700 Network Definition Language (NDL) as implemented on the Burroughs B 1800/B 1700. The specifications contained in this manual describe all of the required and optional statements of the language which are accepted by the B 1800/B 1700 NDL Compiler.

RELATED DOCUMENT

The following document is referenced in this manual:

B 1800/B 1700 Systems System Software Operational Guide, form number 1068731.

SECTION 2

GENERAL NDL INFORMATION

DESCRIPTION OF THE NETWORK DEFINITION LANGUAGE

The B 1800/B 1700 Network Definition Language is classed as both a descriptive language and a programming language. It is a high level language for data communications and provides a simple means of generating a B 1800/B 1700 Network Controller. A network is defined by specifying the network attributes in NDL source code, which includes the physical devices in the network, the line disciplines to be used, the order and priority of line use and the grouping of stations into files.

If the physical data communications network is reconfigured, the user can easily change the attributes that describe the network and regenerate the Network Controller by recompiling.

NDL can accommodate one or more optional Message Control Systems. A Message Control System (MCS) is written by the user when certain functions and system decisions need to be controlled.

The Network Controller

The Network Controller (NC) is the heart of the data communications system, and its function is to process and supervise the flow of messages between application programs and the remote network (stations). The B 1800/B 1700 NDL Compiler translates the source code, and from this produces the Network Controller codefile and the Network Information File (NIF). Figure 2-1 illustrates the NDL generation process.

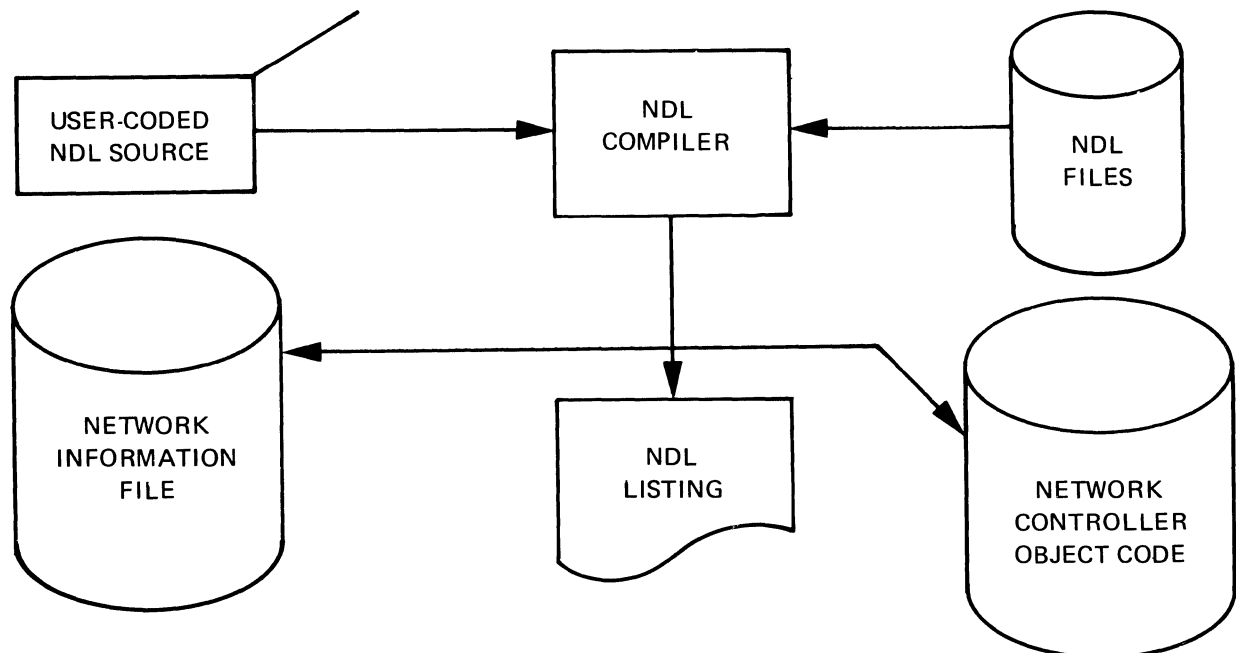


Figure 2-1. NDL Generation Process

The Network Information File contains tables that describe the physical and logical attributes of the network. The initialization values for the Network Controller's LINE, STATION, TERMINAL, and FILE Tables are contained in the NIF.

The Network Controller codefile includes procedures called REQUESTS and CONTROLS that are incorporated as subroutines.

The Network Controller is truly a data communications operating system that executes concurrently with the MCP, and allows efficient multiprogramming of data communications functions with on-site production work. It is a freestanding, normal state program that performs the line discipline, queing, buffer management, audit and reconfiguration functions required by data communications systems. The Network Controller may be considered to be a user-defined extension of the operating system (MCP).

Through the NDL-generated Network Controller, the application program can deal with communications devices in the same manner as with more conventional peripheral devices (card readers, printers, etc.). See Figure 2-2.

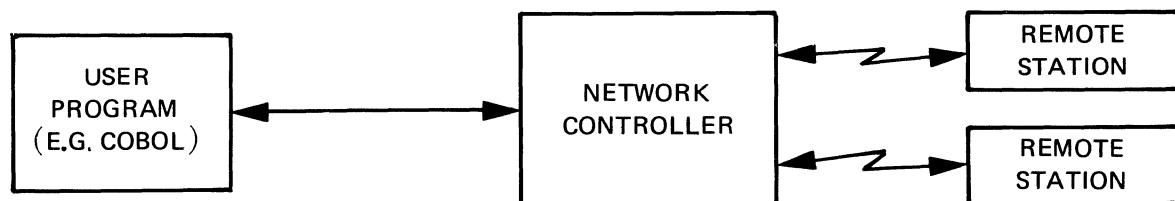


Figure 2-2. The Network Controller

Some of the functions of the Network Controller are:

- a. Provide the data communications line disciplines for all stations.
- b. Construct all of the input or output descriptors for the data communications hardware controls on the B 1800/B 1700.
- c. Initiate I/O and the servicing of I/O completions.
- d. Grants a station the exclusive use of a line.
- e. Make lines or stations ready or not ready.
- f. Provide any optional MCS with appropriate information for it to perform its functions.

The Network Controller performs its functions through the use of user-supplied information and system-supplied routines. The user information is provided by the NDL source program and concerns lines, stations, terminals, LINE CONTROL, REQUEST line disciplines, and interrelated aspects. The system-supplied routines provide for I/O initiation, I/O completion, entrance to user LINE CONTROL and REQUEST line disciplines, communications with the application programs and the MCS, and general supervision of all lines and stations.

When a message is received from a station the Network Controller queues it for the MCS or the application program, and when a message is received from an application program or the MCS it initiates the appropriate procedures to transmit the message to the station. The NC also takes care of simultaneous I/O, multiple lines, recovery from line errors, and auditing, if required.

A B 1800/B 1700 Network Controller will operate asynchronously with the Master Control Program (MCP), providing efficient multiprogramming of data communications functions with on-site batch production jobs.

Message Control System

Some data communications users may want to participate in or rigidly control certain system functions and decisions such as file control, error handling, security, pre-processing, etc. Optional, user-written Message Control Systems can be accommodated by NDL for this purpose. The MCS is written in B 1800/B 1700 COBOL or User Programming Language (UPL). The MCS is more powerful than a normal application program and operates closely with the Network Controller as well as with the MCP.

Burroughs supplies several “illustrative” Message Control Systems named MCSI and MCSII. The intent of these MCS’s is to demonstrate some of the techniques required for writing a Message Control System. These “illustrative” MCS’s are not intended to be standard MCS’s, and no support of them is intended or implied.

TYPES OF DEFINITIONS

Three different aspects of the remote network may be described by the Network Definition Language (NDL).

- a. The Line Discipline definitions include a set of specified procedures called REQUESTS which indicate how to communicate with the remote network.
- b. The Physical Network definition of NDL is used to define the network in terms of number of lines, line attributes, number of stations, and station attributes.
- c. File definitions allow sets of stations to be grouped into logical entities called files. File names are used by object programs to identify the stations or set of stations with which they communicate.

NDL SOURCE PROGRAM ORGANIZATION

Every NDL source program must contain these seven sections in the following order:

DECLARATION
REQUEST
LINE CONTROL
TERMINAL
STATION
LINE
FILE

The physical and logical specifications of a data communications network may be described in NDL by these seven sections. The order in which statements must appear within the sections is significant only within the REQUEST and LINE CONTROL sections. A brief description of each section and its functions follow.

DECLARATION Section – This section is optional. It provides a means of specifying global definitions for the user’s Network Controller. Constants, auditfiles, and NIF file name are defined in this section.

REQUEST Section – The line discipline routines (REQUESTS) used by the various terminals of the remote network are defined by the user in this section. A REQUEST must be defined for each terminal. The REQUEST Section is a higher-level language and contains constructs such as those in the following example.

```

TRANSMIT EOT, ADDRESS, POL, ENQ.
FINISH TRANSMIT – INITIATE RECEIVE.
IF EXCEPTION THEN
DO.
    RETRY := RETRY-1.
    IF RETRY = 0 THEN TERMINATE ERROR.
    ELSE UNDO REQUESTLOOP.
END.
FETCH CHAR.
IF CHAR = SOH THEN .... ETC.

```

LINE CONTROL Section – This section is similar to the REQUEST Section and allows for conditional statements. The unconditional statements are different, however, and a partial list of them follows:

```

INITIATE AUTOPOLL
CONTINUE
POLL
UNDO

```

As the name implies, this section's function is to control the line and allocate the line to different tasks. For example, if a line has a high priority station the LINE CONTROL Section has the ability to pre-empt outputs to other stations. It can also create a poll string and then INITIATE AUTOPOLL, as in this example:

```

IF LINE (TOG [0]) THEN
DO CREATEPOLL FOREVER.
    STATION := STATION + 1.
    IF STATION (VALID) THEN POLL EOT, ADDRESS, POL, ENQ.
    IF ENDOFBUFFER THEN UNDO CREATEPOLL.
    IF STATION = MAXSTATION THEN UNDO CREATEPOLL.
END CREATEPOLL.
INITIATE AUTOPOLL.

```

TERMINAL Section – The user describes the physical hardware characteristics of the different types of remote devices in the network in this section.

STATION Section – This section specifies the type of terminal, the station's line address (if any), and information about the usage of the station.

LINE Section – The LINE Section provides the user with the means to describe the logical and physical characteristics of the lines in the system.

FILE Section – This section provides a means of associating stations with object program remote files.

When the NDL source program is compiled, the REQUEST and LINE CONTROL Sections produce executable object code which is merged with the Skeletal Network Controller to produce the Network Controller. The descriptive sections of NDL are used to create the Network Information File (NIF) and supply the information for the tables referred to previously in this document.

A detailed illustration of the NDL generation process appears in Figure 2-3.

Additional information concerning each of the seven sections of the NDL source code appears in the semantics portion of the sections of this manual entitled Syntax and Semantics of Sections.

The language processor will check for contradictory statements and/or missing statements at the end of every specification. After each section defining terminals, stations, lines, and files has been scanned, a summary of any errors or incomplete statements is printed.

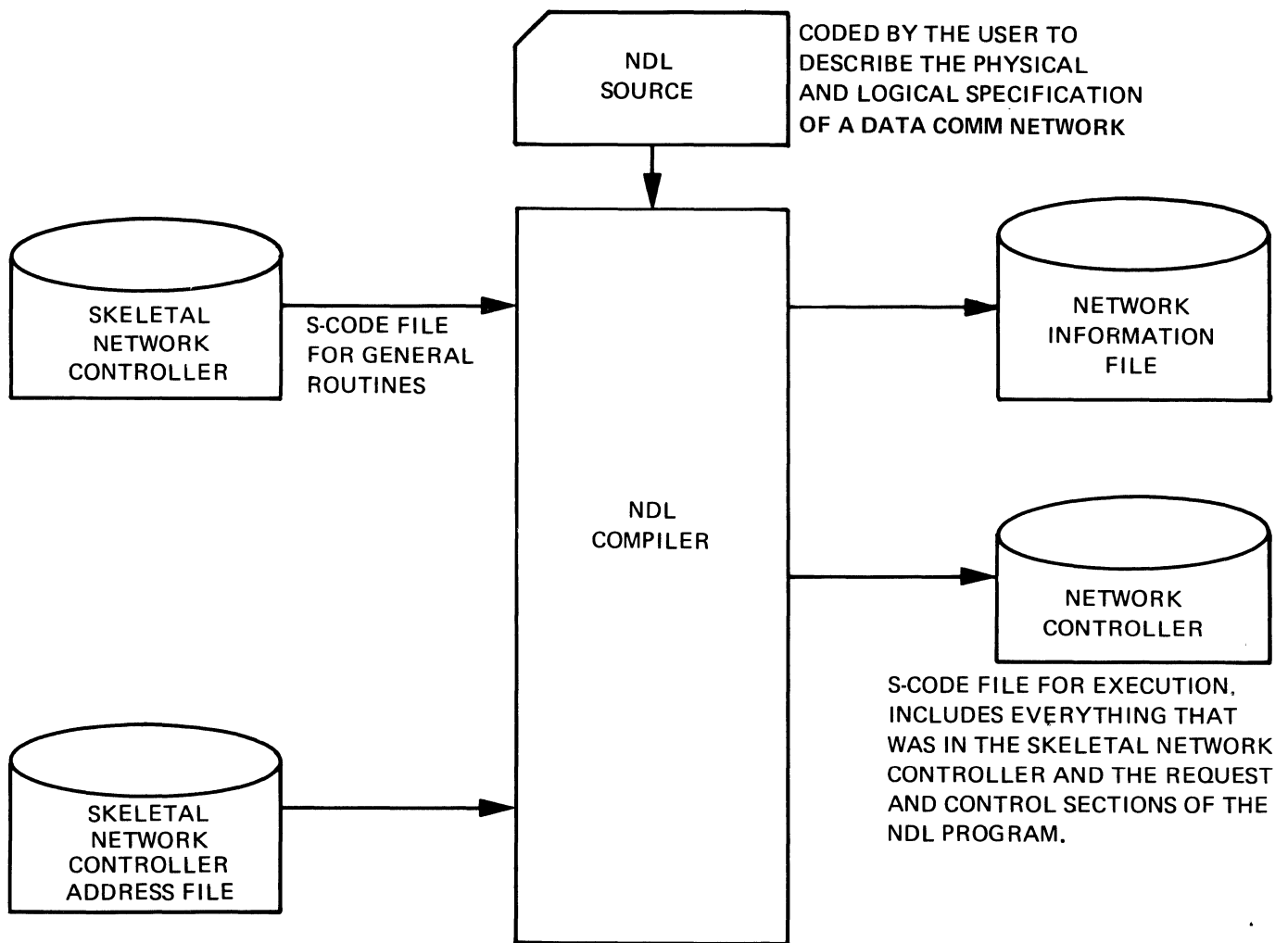


Figure 2-3. Detailed NDL Generation Process

The NDL Compiler will verify that the rules presented in this manual have been followed and will generate an object program in machine code that is ready to be executed.

STATEMENT REQUIREMENTS AND DEFAULTS

Requirements

Because the various remote devices possess different characteristics, it is possible that the required statements in a description will vary from one definition to another. NDL is summarized in the following observations about the statements making up a description:

- a. Some of the attributes defined in the syntax are required.
- b. Some of the attributes are dependent on the presence of other attributes.
- c. Some of the attributes defined in the syntax are optional.

The interdependency of the various attributes will be pointed out in the semantics sections of the applicable statements.

Defaults

The NDL Compiler and Data Comm system as a whole make no assumptions for any part of a network. The default lists that are defined in the syntax throughout this manual provide the user with a method to conveniently group sets of attributes for terminals, stations, lines, and files under one or more identifiers.

Example:

```
TERMINAL TD700:
  REQUEST=POLLTCTD:RECEIVE,FASTSELTC: TRANSMIT.
  BUFFERSIZE = 270.
  ADDRESS = 2.
STATION DEFAULT DF :
  MYUSE = INPUT, OUTPUT.
  TERMINAL = TD700.
  RETRY = 5,
STATION S1 :
  DEFAULT = DF .
  ADDRESS = "S1" :
STATION S2 :
  DEFAULT = DF .
  ADDRESS = "00" .
```

The NDL Compiler does not check for contradictory statements within a default definition. The attributes specified within a description always override those listed in a default definition when conflicts occur.

SECTION 3

STRUCTURE OF THE LANGUAGE

SYNTAX CONVENTIONS

The Backus-Naur Form (BNF) is a metalanguage and is used to describe the syntax of the B 1800/B 1700 Network Definition Language.

Metalinguistic Symbols

The BNF uses only four metalinguistic symbols to differentiate between the symbols of the metalanguage and those of the language being described. Left and right broken brackets are considered to be one symbol, as are left and right braces.

The metalinguistic symbols used in this document with a description of them follows:

- ⟨ ⟩ Left and right angle brackets are used to contain one or more digits and/or letters representing a metalinguistic variable whose definition is given by a metalinguistic formula.
- :: = This symbol means “is defined as.” It separates the BNF variable on the left side of a BNF formula from the syntactic definition on the right side.
- | The symbol “ | ” means exclusive “or.” It separates multiple definitions of a BNF variable. The following is an example of the use of this symbol (verticle bar).

⟨ population ⟩ :: = ⟨ fixed-population ⟩ | ⟨ variable-population ⟩

This BNF formula would read: a population is defined as a fixed population or a variable population.

- { } Braces are used to enclose BNF variables which are defined by the meaning of the English language expression contained within the braces. This formulation is used only when it is impossible or impractical to use a BNF formula. An example using this symbol follows:

⟨ S-code ⟩ :: = { an integer m such that $0 \leq m \leq 99$ }

This formula would read as follows: an S-code is defined as an integer m, so that zero is less than or equal to m which is less than or equal to 99.

BASIC ELEMENTS

Character Set

The character set for NDL is drawn from the Extended Binary Coded Decimal Interchange Code (EBCDIC) character set. Any EBCDIC character is valid in an ⟨ EBCDIC STRING ⟩. The Data Comm system is internally EBCDIC. All characters are translated when appropriate (that is, when transmission is to a non-EBCDIC device).

Syntax:

⟨ LETTER ⟩ :: = A | B | C | D | E | F | G | H | I | J | K | L | M |
N | O | P | Q | R | S | T | U | V | W | X | Y | Z

$\langle \text{DIGIT} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

$\langle \text{SPECIAL CHARACTER} \rangle ::= + \mid - \mid \cdot \mid \cdot \mid \cdot \mid : \mid [\mid] \mid < \mid > \mid \$ \mid * \mid \% \mid @ \mid " \mid (\mid) \mid \leq \mid \geq \mid = \mid \langle \text{SLASH} \rangle \mid \langle \text{VERTICAL BAR} \rangle \mid \langle \text{SPACE} \rangle$

$\langle \text{SLASH} \rangle ::= /$

$\langle \text{VERTICAL BAR} \rangle ::= |$

$\langle \text{SPACE} \rangle ::= (\text{ONE GRAPHIC SPACE, HEXADECIMAL 40})$

Semantics:

The character set for NDL is a subset of the EBCDIC character set containing letters, digits, special characters, and the space.

Basic Components

Basic components are the most primitive structures of the NDL language.

Syntax:

$\langle \text{BASIC COMPONENT} \rangle ::= \langle \text{IDENTIFIER} \rangle$
 $\quad \mid \langle \text{INTEGER} \rangle$
 $\quad \mid \langle \text{STRING} \rangle$
 $\quad \mid \langle \text{OPERATOR} \rangle$
 $\quad \mid \langle \text{LOGICAL VALUE} \rangle$
 $\quad \mid \langle \text{BRACKET} \rangle$
 $\quad \mid \langle \text{SEPARATOR} \rangle$
 $\quad \mid \langle \text{VARIABLE} \rangle$

IDENTIFIERS

Syntax:

$\langle \text{IDENTIFIER} \rangle ::= \langle \text{LETTER} \rangle \mid \langle \text{IDENTIFIER} \rangle \langle \text{LETTER} \rangle \mid \langle \text{IDENTIFIER} \rangle \langle \text{DIGIT} \rangle$

Restrictions:

- a. An identifier must begin with a letter.
- b. Special characters may not be embedded within the identifier.
- c. There is no maximum identifier length, but only the first 10 characters will be used by the compiler.
- d. Identifiers must be unique.

Semantics:

Identifiers have no intrinsic meaning. They are used to name labels, constants, requests, terminals, stations, lines, and files.

INTEGERS

Syntax:

⟨ INTEGER ⟩ ::= ⟨ DIGIT ⟩ | ⟨ INTEGER ⟩ ⟨ DIGIT ⟩

Semantics:

Only positive integers are allowed in NDL, and they are loaded as 24-bit binary numbers.

STRINGS

Syntax:

⟨ STRING ⟩ ::= ⟨ HEXADECIMAL STRING ⟩ | ⟨ EBCDIC STRING ⟩

⟨ HEXADECIMAL STRING ⟩ ::= 4 “ ⟨ HEXADECIMAL CHARACTER CONCATENATION ⟩ ”

⟨ HEXADECIMAL CHARACTER CONCATENATION ⟩ ::= ⟨ HEXADECIMAL CHARACTER ⟩
⟨ HEXADECIMAL CHARACTER ⟩ | ⟨ HEXADECIMAL CHARACTER CONCATENATION ⟩
⟨ HEXADECIMAL CHARACTER ⟩ ⟨ HEXADECIMAL CHARACTER ⟩

NOTE

Hexadecimal characters are added by 2's.

⟨ HEXADECIMAL CHARACTER ⟩ ::= 0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F

⟨ EBCDIC STRING ⟩ ::= “ ⟨ EBCDIC CHARACTER CONCATENATION ⟩ ”

⟨ EBCDIC CHARACTER CONCATENATION ⟩ ::= ⟨ EBCDIC CHARACTER ⟩ | ⟨ EBCDIC CHARACTER ⟩
⟨ EBCDIC CHARACTER CONCATENATION ⟩

Semantics:

The maximum length of a ⟨ HEXADECIMAL STRING ⟩ is 40 bits, or 10 hexadecimal digits.

Any EBCDIC character may appear in an ⟨ EBCDIC string ⟩. An internal quote is represented by two consecutive quotes. For example, “””” represents an EBCDIC string of one single quote.

OPERATORS

Syntax:

⟨ OPERATOR ⟩ ::= ⟨ UNARY OPERATOR ⟩ | ⟨ BINARY OPERATOR ⟩

⟨ UNARY OPERATOR ⟩ ::= + | - | NOT

⟨ BINARY OPERATOR ⟩ ::= ⟨ ARITHMETIC OPERATOR ⟩
| ⟨ LOGICAL OPERATOR ⟩
| ⟨ RELATIONAL OPERATOR ⟩

⟨ ARITHMETIC OPERATOR ⟩ ::= + | - | * | ⟨ SLASH ⟩

⟨ LOGICAL OPERATOR ⟩ ::= AND | OR | XOR

⟨ RELATIONAL OPERATOR ⟩ ::= EQ | NE | GT | GE | LT | LE | = | > | <

Semantics:

The alphabetic relational operators are abbreviations having the following meanings:

<u>Relational Operator</u>	<u>Meaning</u>
EQ	Equal
NE	Not equal
GT	Greater than
GE	Greater than or equal
LT	Less than
LE	Less than or equal

Operators are used in expressions (see Expressions). Unary operators require one operand, as in NOT BREAK. Binary operators require two operands, as in "10 * 3."

LOGICAL VALUES

Syntax:

⟨ LOGICAL VALUE ⟩ ::= TRUE | FALSE

Semantics:

TRUE is equivalent to the integer 1.

FALSE is equivalent to the integer 0.

Example:

The expression BREAK EQ TRUE could also be written BREAK EQ 1.

BRACKETS

Syntax:

⟨ BRACKET ⟩ ::= () | []

Semantics:

Parentheses signify a modification or variant form of the preceding symbol.

Examples:

TRAN (TRANSMIT)
TRAN (RECEIVE)

Brackets ([]) are used for subscripts of TOG and TALLY.

Examples:

TOG [1]
TALLY [0]

SEPARATORS

Syntax:

⟨ SEPARATOR ⟩ ::= , | . | : | ⟨ SPACE ⟩

⟨ SPACE ⟩ ::= ⟨ BLANK ⟩ | ⟨ SPACE ⟩ ⟨ BLANK ⟩

Semantics:

⟨ BASIC COMPONENTS ⟩ may be separated by one or more blanks or spaces, a comma (,), period (.), or colon (:).

The following examples are equivalent:

1. MYUSE = INPUT, OUTPUT.
2. MYUSE (b) (b) = (b) (b) (b) INPUT (b) , (b) (b) OUTPUT (b) .
3. MYUSE
 =
 INPUT ((b) - indicates a blank space.)
 ,
 OUTPUT
 .

VARIABLES

Syntax:

⟨ VARIABLE ⟩ ::= ⟨ SYSTEM STATUS VARIABLE ⟩ | ⟨ ERROR FLAG ⟩ | ⟨ USER TOGGLE ⟩ |
 ⟨ USER TALLY ⟩ | ⟨ STRING FUNCTION ⟩

Semantics:

All variables can be referenced in REQUESTS and CONTROLS. Many of them are assignable and can have their values changed. Table 3-1, Characteristics of Variables, shows which NDL variables are assignable.

SYSTEM STATUS VARIABLES

Syntax:

```
< SYSTEM STATUS VARIABLE > :: = CHAR  
| CHARACTER  
| FREQUENCY (INPUT)  
| FREQUENCY (OUTPUT)  
| IODESC  
| INPUTATTACHED  
| OUTPUTATTACHED  
| LENGTH (INPUT)  
| LENGTH (OUTPUT)  
| LINE  
| LINE (CONTROL KEY)  
| LINE (QUEUED)  
| MAXSTATIONS  
| RESULTDESC  
| RETRY  
| SEQUENCE  
| STATION  
| STATION (ENABLED)  
| STATION (LINE)  
| STATION (MYUSE)  
| STATION (QUEUED)  
| STATION (READY)  
| STATION (TYPE)  
| STATION (VALID)  
| TERMINALTYPE  
| TIME  
| TRAN (RECEIVE)  
| TRAN (TRANSMIT)
```

Semantics:

The NDL programmer must use a set of predefined identifiers in the REQUEST and CONTROL Sections to test or set the value of system items. The user may define constants in a DECLARATION Section.

System status variables refer to items in the Network Controller's tables, except for the TIME variable which is an MCP communicate.

All of the system status variables can appear in an expression in either the REQUEST or CONTROL Sections. Some of the system status variables are assignable and can have their values changed explicitly in a REQUEST or CONTROL, while the remainder are read-only. Table 3-1, Characteristics of Variables, shows which system status variables are assignable.

A description of each of the system status variables and some examples of their use follows:

CHAR, CHARACTER

Both CHAR and CHARACTER refer to the character register. They are intended to be used for processing a message character by character. Both variables are assignable.

Example:

```
INITIALIZE TEXT.  
DO ECHOLOOP FOREVER.  
  FETCH CHAR.  
  TRANSMIT CHAR.  
  IF CHAR EQ ETX THEN UNDO.  
END ECHOLOOP.
```

The FETCH statement is used to load the register from the input message.

FREQUENCY (INPUT), FREQUENCY (OUTPUT)

These are both read-only items in the STATION table. The < FREQUENCY STATEMENT > in the STATION Section is used to declare both of these variables.

IODESC

This is a read-only field in the LINE table. IODESC contains the two 144-bit I/O descriptors used for I/O on the current line. It is useful in debugging in the < AUDIT STATEMENT >.

INPUTATTACHED

This is an entry in the LINE Table. It is not assignable. It is set TRUE if there is a valid input message buffer into which data may be received, and FALSE if no input message buffer is attached to the line.

OUTPUTATTACHED

This is an entry in the LINE table. It is not assignable. It is set TRUE if there is a valid output message buffer attached to the line for transmission, and FALSE if there is no output message buffer attached.

LENGTH (INPUT), LENGTH (OUTPUT)

Each of these is a read-only field in the LINE table. They are used to indicate the length in bytes of the input or output text respectively. The length of the message header or ETX is not included.

LINE

The current line number is stored in this read-only field of the LINE table.

LINE (CONTROL KEY)

This is a read-only field in the LINE table whose function is to drive the “flow of control” in the CONTROLS, usually as a CASE Statement index (refer to the CONTROL Section). The value of LINE (CONTROL KEY) indicates the reason why the Network Controller entered the CONTROL, and also dictates the action that the CONTROL should take. The possible values and their meanings for LINE (CONTROL KEY) are as follows:

<u>Value</u>	<u>Meaning</u>
0	The line is now idle. I/O can be initiated.
1	A change in the status of a station or stations has occurred on this line due to one of the following conditions: <ol style="list-style-type: none">This was the first time through the CONTROL.A remote file OPEN or CLOSE has occurred involving a station on this line.The Network Controller has initiated End Of Job. All stations are disabled for input. The CONTROL should rebuild the poll list (if there is one,) or retest the stations on the line. If no stations are eligible for I/O, the CONTROL executes an INITIATE IDLE.
2	When LINE(CONTROL KEY) = 2, there are two possible conditions that are described below: <ol style="list-style-type: none">An output message has been queued for this line, but a READ is now in process with no TIMEOUT or AUTOPOLL. The CONTROL must do one of the following:

<u>Value</u>	<u>Meaning</u>
2 (cont)	<ol style="list-style-type: none"> 1. INITIATE CANCEL. This terminates the I/O operation. 2. CONTINUE. This permits the I/O operation to continue, and leaves output messages, if any, in the queue. b. An I/O has been in progress for more than 30 seconds, and that I/O is either a READ with NO TIMEOUT, a WRITE/READ with NO TIMEOUT, or an AUTOPOLL.
3	A "ring complete" has been received. This value is used for switched-line Data Comm environments only.
4	The last REQUEST on this line did a TERMINATE OUTPUT (RETURN). The CONTROL must eventually execute a CONTINUE to restart that REQUEST. At this point the CONTROL may change the value of STATION before returning, as might be desired in some Remote Job Entry (RJE) line disciplines. For example, a remote computer's output device could be changed from line printer to console printer for the next block of data.
5	<p>The CONTROL is entered with an output message queued and:</p> <ol style="list-style-type: none"> a. The line is not currently connected, and b. An ACU adapter is available to connect the line as indicated by a strap in the regular adapter, and c. A telephone number is provided.

LINE (QUEUED)

This is an entry in the LINE table. It is a read-only field and is set TRUE if a message is queued for output for any station on this line.

MAXSTATIONS

The maximum number of stations on the current line, including dummy stations, is indicated in this field. It is a read-only field. MAXSTATIONS is normally used to test the current value of STATION to determine if it should be reset to one.

Example:

```
IF STATION EQ MAXSTATIONS
  THEN STATION := 1.
ELSE STATION := STATION + 1.
```

RESULTDESC

RESULTDESC references sub-fields of IODESC, and is useful in debugging a data communications system. It is a read-only field.

Example:

```
AUDIT TRACEFILE (RESULTDESC).
```

RETRY

RETRY is a tally field used to count RECEIVE and TRANSMIT retries in REQUESTS. It is assignable in both REQUESTS and CONTROLS. The initial value of RETRY is set by the (RETRY STATEMENT) in the STATION Section. It may be reset to this value at run-time by INITIALIZE RETRY in REQUESTS. RETRY is never implicitly reinitialized by the Network Controller. After each unsuccessful RECEIVE or TRANSMIT attempt, a one may be subtracted from RETRY. When RETRY equals zero, a TERMINATE ERROR would be appropriate, as shown in the following example.

Example:

```
RETRY := RETRY - 1
IF RETRY EQ 0 THEN TERMINATE ERROR.
```

SEQUENCE

This system status variable refers to the sequence number of the NDL source code line in which it appears. SEQUENCE is intended for use with the DISPLAY and the AUDIT statements.

Example:

```
DISPLAY "ERROR DETECTED AT", SEQUENCE.
```

STATION

This is a field in the LINE table that is assignable only by CONTROLS. The value of STATION must always be in the range: $1 \leq \text{STATION} \leq \text{MAXSTATIONS}$. If STATION is allowed to exceed these bounds a fatal run error will occur, so the programmer should always test the value of STATION before changing it.

STATION numbers are determined by the order of stations in the <LINE STATION STATEMENT> in the LINE Section. Note that STATION refers only to the line-relative station number. It is not the same as the LOGICAL STATION NUMBER which appears in the 50-byte message header. LOGICAL STATION NUMBER is unique to each station in the network, no matter which line it is on, and is determined by the order in which stations are declared in the STATION Section.

STATION is also not the same as the FILE-RELATIVE STATION NUMBER used as a key by user programs in doing I/O on remote files. Neither LOGICAL STATION NUMBER or FILE-RELATIVE STATION NUMBER may be referenced within NDL.

STATION (ENABLED)

This is an entry in the STATION table that is a read-only field. It must be set TRUE for an input message to be received from the current station. If it is FALSE, an INITIATE INPUT will cause a run error. STATION (ENABLED) is set TRUE if an MCS is present or a program has opened an input file that references the station. STATION (ENABLED) is set FALSE if the remote input file for that station is full. In that case, no input operations may be performed until STATION (ENABLED) is set TRUE.

STATION (LINE)

STATION (LINE) is a read-only entry in the STATION table. It is used in a switched-line Data Comm environment to indicate the line index of the line currently associated with the station.

STATION (LINE) must be used in a LINE CONTROL to determine if a station is assigned to the current line, another line, or unassigned (zero).

The field is initialized when a LINE CONTROL executes an INITIATE (INPUT, OUTPUT, INPUTOUTPUT, or OUTPUTINPUT). It is reset to zero when a CONTROL INITIATE DISCONNECT is executed in a LINE CONTROL, or when a TERMINATE DISCONNECT or TERMINATE RELEASE STATION are executed in a REQUEST.

STATION (MYUSE)

This is an item in the STATION table that is read-only. The value of this field is set by the MYUSE statement in the STATION Section.

The possible values of STATION (MYUSE) and what they represent follows:

<u>Value</u>	<u>Meaning</u>
1	STATION is for INPUT use only.
2	STATION is for OUTPUT use only.
3	STATION is used for INPUT and OUTPUT. A three is the default value.

STATION (QUEUED)

An item in the STATION table, STATION (QUEUED) is read-only. If the current station has at least one message in its output queue STATION (QUEUED) will be TRUE. If it is FALSE an INITIATE OUTPUT will be ignored.

STATION (READY)

This is an entry in the STATION table that is read-only. It must be TRUE in order to INITIATE INPUT or INITIATE OUTPUT. It can be set TRUE or FALSE by the MCS. A TERMINATE ERROR will set the bit FALSE (NOT READY), only if an MCS is present.

STATION (TYPE)

This variable should be used to build poll strings in order to take advantage of group poll (refer to the POLL statement in the CONTROL Section). It is a read-only field, and indicates the MASTER/SLAVE TYPE of the remote device currently indexed by STATION.

The possible values and what they represent follow:

<u>Value</u>	<u>Meaning</u>
1	SLAVE STATION.
2	CONTROLLER STATION.
3	NORMAL STATION.

The value of this field is set by the CONTROLLER statement in the STATION Section.

STATION (VALID)

This is a read-only entry in the STATION table. It will be TRUE if there is a real station (a valid station) on the current line corresponding to the current station number.

TERMINALTYPE

This is a read-only field in the TERMINAL table. The value is set by the `<TERMINAL TYPE STATEMENT>` in the TERMINAL Section. The following example would apply if the terminal were a TELETYPE (TTY).

Example:

```
TERMINAL TTY:  
  REQUEST = TTYECHO:RECEIVE.  
  MAXINPUT = 256.  
  TYPE     = 6.
```

Refer to the TERMINAL Section for the list of hardware terminal types and their associated values.

TIME

The function of this variable is to make available the time of day in tenths of seconds. It is assignable in REQUESTS. When TIME is invoked during program execution, the value is loaded from the system clock, and is assigned to TIME (TALLY) for message headers.

Example:

```
TIME (TALLY) := TIME.
```

TRAN (RECEIVE)

The function of this field of the STATION table is to hold the “transmission number” of input messages. The length is set in the TERMINAL Section by the `<TRANSMISSION NUMBER STATEMENT>`. TRAN (RECEIVE) is assignable and is normally loaded from the internal message header by the RECEIVE TRAN statement. The field is set to zero by the INITIALIZE TRAN (RECEIVE) statement. In the following example, the field length would be set to 2 bytes in the TERMINAL Section.

Example:

```
TRANSMISSION = 2.
```

TRANERR will be set by the Network Controller after execution of RECEIVE TRAN if the new transmission number equals the old transmission number.

TRAN (TRANSMIT)

The output message’s transmission number is stored in this field of the STATION table. The length of the field is set by the TRANSMISSION NUMBER statement in the TERMINAL Section. The Network Controller resets TRAN (TRANSMIT) to zeroes at BOJ time, so it should be set to some other value before sending it. Normally, it should be given a value different from the value of the last output transmission number, either by the REQUEST or by a Message Control System. An example of a direct assignment in a REQUEST follows:

Example:

TRAN (TRANSMIT) := DECIMAL (TALLY [0], 3).

ERROR FLAGS

Syntax:

⟨ ERROR FLAG ⟩ ::= ⟨ RECEIVE ERROR FLAG ⟩
 | ⟨ TRANSMIT ERROR FLAG ⟩
 | ⟨ ITEM ERROR FLAG ⟩

⟨ RECEIVE ERROR FLAG ⟩ ::= BREAK | ACCESSERR | EXCEPTION
 | LOSSOFCARRIER | PARITY
 | TIMEOUT

⟨ TRANSMIT ERROR FLAG ⟩ ::= BREAK | EXCEPTION
 | LOSSOFDSR | TIMEOUT

⟨ ITEM ERROR FLAG ⟩ ::= ADDERR | ENDOFBUFFER
 | FORMATERR | TRANERR

Semantics:

All of the Error Flags are assignable and accessible in REQUESTS. In addition, ENDOFBUFFER is assignable and accessible in CONTROLS also. The Network Controller resets all Error Flags before initiating an I/O operation. Each of the error Flags is set automatically on the occurrence of the appropriate error condition. In input REQUESTS the Error Flags are moved to the read-error field in the 50-byte internal message header. The Error Flag identifiers and their descriptions follow:

<u>Error Flag</u>	<u>Description</u>
ADDERR	This flag is set when a message is input from a station and an error in the station address is detected. The size in bytes and the value of the station address as defined in the ⟨ TERMINAL DEFINITION ⟩ and ⟨ STATION ATTRIBUTE LIST ⟩ allow the Network Controller to check the correct value against that which is received.
BREAK	A BREAK signal was received from a remote station during the output of a message to that station.
ACCESSERR	This flag is set if the MLC/SLC was not granted memory access before the next character was received on the line. The previous character is destroyed as a result.
ENDOFBUFFER	May be set by these conditions: <ol style="list-style-type: none">1. In an input REQUEST, the number of characters stored in the buffer exceeds the maximum specified by the ⟨ BUFFERSIZE STATEMENT ⟩ in the ⟨ TERMINAL DEFINITION ⟩.2. In an output REQUEST, the transmit I/O reached the end of the message and no terminating character was encountered.3. In an input REQUEST, the number of fetched characters exceed the number actually in the buffer.

<u>Error Flag</u>	<u>Description</u>
	4. In the CONTROL Section if the AUTOPOLL buffer overflows.
EXCEPTION	Will be set if ACCESSERR, BREAK, ENDOFBUFFER, LOSSOFCARRIER, LOSSOFDSR, PARITY, or TIMEOUT is set because of an initiated I/O.
FORMATERR	Will be set if a REQUEST uses the RECEIVE < STRING > option to check input data, and the characters received do not correspond to the < STRING > specified.
LOSSOFCARRIER	This flag indicates that loss of carrier frequency was reported by the data set.
LOSSOFDSR	Is set if the hardware reports that data set READY was dropped during an I/O.
PARITY	When a parity or Block Check Character (BCC) error is detected while receiving a message this flag will be set.
TIMEOUT	Is set when a character was not received during transmission of an input message, or when the first character was not transmitted within the time specified by the adapter.
TRANERR	Will be set following RECEIVE TRAN (RECEIVE) when the transmission number received was equal to the transmission number in the last message from the same station.

USER TOGGLES

Syntax:

```

< USER TOGGLE > :: = TOG [ < TOG INDEX > ]
                    | LINE(TOG [ < TOG INDEX > ] )
< TOG INDEX >    :: = < integer > | < variable >

```

Semantics:

TOG [< TOG INDEX >]

By the use of user toggles, a variable number of one-bit fields in each STATION table and in the header of each message are reserved for the program. These fields are referred to as toggles, and are assignable and accessible in both REQUESTS and CONTROLS. The toggles are more useful in REQUESTS because there can be one set of toggles for each station. Toggles are useful for local data or for communicating with an optional Message Control System.

Toggles can be set by an ASSIGNMENT statement or an INITIALIZE statement. Toggles are automatically loaded in the 50-byte internal message header of an input message when a TERMINATE INPUT statement is executed in a REQUEST. The toggles can also be loaded from the message header of an output message by the INITIALIZE TOGGLE statement. When TERMINATE ERROR is executed, the toggles are loaded to the 50-byte internal message header.

LINE(TOG [< TOG INDEX >])

By the use of user toggles, a variable number of one-bit fields in the LINE table are reserved for the program. These fields, referred to as toggles, are assignable and accessible in both REQUESTS and CONTROLS.

Restrictions:

- a. There must be a MAX TOG statement in the DECLARATION Section for any toggle that is indexed by a variable.
- b. A maximum of 100 line toggles and 100 station toggles can be used in an NDL program.
- c. Toggle indexes start at 0 and proceed through ninety-nine (99).
- d. If a MAX TOG statement is not declared, the number of toggles reserved will be determined by the largest index used in the user's NDL program.

USER TALLIES

Syntax:

```
<USER TALLY> :: =    TALLY [ <TALLY INDEX> ]  
                    | LINE (TALLY [ <TALLY INDEX> ] )  
                    | TIME (TALLY)
```

```
<TALLY INDEX> :: = <integer> | <variable>
```

Semantics:

TALLY [<TALLY INDEX>]

A variable number of eight-bit fields in each STATION table and in the message header of each message can be reserved for the programmer's use. These fields are referred to as tallies, and are assignable and accessible in REQUESTS and CONTROLS. Tallies are useful for local data or for communicating with an optional Message Control System.

Tallies can be set by an ASSIGNMENT statement or an INITIALIZE statement. Tallies are automatically loaded in the 50-byte internal message header of an input message when a TERMINATE INPUT statement is executed in a REQUEST, and can also be loaded from the message header of an output message by the INITIALIZE TALLY statement. When TERMINATE ERROR is executed, the tallies are loaded from the 50-byte internal message header.

LINE(TALLY [<TALLY INDEX>])

A variable number of eight-bit fields in each LINE table and in the message header of each message can be reserved for the programmer's use. These fields are referred to as line tallies, and are assignable and accessible in REQUESTS and CONTROLS.

TIME(TALLY)

This is a 20-bit field in the STATION table that is assignable and accessible only in REQUESTS. It is normally assigned the value of TIME when used in input REQUESTS. When a RECEIVE TEXT statement is executed, the Network Controller moves the contents of this field to the internal message header of the message.

Restrictions:

- a. There must be a MAX TALLY statement in the DECLARATION Section for any tally that is indexed by a variable.
- b. A maximum of 100 line tallies and 100 station tallies can be used in an NDL program.
- c. Tally indexes start at 0 and proceed through 99. Ninety-nine (99) is the largest allowable index.
- d. If a MAX TALLY statement is not declared, the number of tallies reserved will be determined by the largest index used in the user's NDL program.

STRING FUNCTIONS

Syntax:

$\langle \text{STRING FUNCTION} \rangle ::= \langle \text{DECIMAL FUNCTION} \rangle \mid \langle \text{CONVERT FUNCTION} \rangle$
 $\langle \text{DECIMAL FUNCTION} \rangle ::= \text{DECIMAL} (\langle \text{EXPRESSION} \rangle, \langle \text{INTEGER} \rangle)$
 $\langle \text{CONVERT FUNCTION} \rangle ::= \text{CONVERT} (\langle \text{EXPRESSION} \rangle)$
 $\quad \mid \text{CONVERT} (\langle \text{EXPRESSION} \rangle, 1)$

Semantics:

The DECIMAL function is used to convert the value of the $\langle \text{EXPRESSION} \rangle$ to an EBCDIC string of decimal digits of length $\langle \text{INTEGER} \rangle$. A maximum of eight characters will be returned, even if the value of $\langle \text{INTEGER} \rangle$ is greater than eight. If the value of the $\langle \text{EXPRESSION} \rangle$ is too great to be represented in 24 bits, only the least significant 24 bits will be used.

Example:

DECIMAL (“FF”, 3) will be converted to the string “255” or F2F5F5.

The CONVERT function converts binary strings of any length to character strings. When “,1” is specified following the (EXPRESSION), a character will be generated for each bit of the result (“0” for a binary 0 or “1” for a binary 1). The most significant bit positions will be zero-filled if necessary, so that a result 24 bits in length will be generated.

Example:

CONVERT (255,1) will result in the following character string: “000000000000000011111111”.

The default usage results in each character representing four bits (hexadecimal conversion).

Example:

CONVERT (255) will result in the following character string: “0000FF”.

Table 3-1. Characteristics of Variables

Variables	Assignable
System Status Variables	
CHAR, CHARACTER	Yes
FREQUENCY (INPUT)	No
FREQUENCY (OUTPUT)	No
IODESC	No
INPUTATTACHED	No
OUTPUTATTACHED	No
LENGTH (INPUT)	No
LENGTH (OUTPUT)	No
LINE	No
LINE (CONTROL KEY)	No
LINE (QUEUED)	No
MAXSTATIONS	No
RETRY	Yes
RESULTDESC	No
SEQUENCE	No
STATION	Yes, by CONTROLS
STATION (ENABLED)	No
STATION (MYUSE)	No
STATION (QUEUED)	No
STATION (READY)	No
STATION (TYPE)	No
STATION (VALID)	No
TERMINALTYPE	No
TIME	No
TRAN (RECEIVE)	Yes
TRAN (TRANSMIT)	Yes

Table 3-1. Characteristics of Variables (Continued)

Variables	Assignable
<p>Error Flags</p> <p>ACCESSERR ADDERR BREAK ENDOFBUFFER EXCEPTION FORMATERR LOSSOF CARRIER LOSSOFDSR PARITY TIMEOUT TRANERR</p>	<p>Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes</p>
<p>User Toggles</p> <p>TOG [< TOG INDEX >] LINE (TOG [< TOG INDEX >])</p>	<p>Yes Yes</p>
<p>User Tallies</p> <p>TALLY [< TALLY INDEX >] LINE(TALLY [< TALLY INDEX >]) TIME(TALLY)</p>	<p>Yes Yes Yes</p>

Expressions

Syntax:

$\langle \text{EXPRESSION} \rangle ::= \langle \text{TERM} \rangle \mid \langle \text{TERM} \rangle \langle \text{BINARY OPERATOR} \rangle \langle \text{EXPRESSION} \rangle$
 $\mid \langle \text{EXPRESSION} \rangle \langle \text{BINARY OPERATOR} \rangle \langle \text{TERM} \rangle$

$\langle \text{TERM} \rangle ::= \langle \text{PRIMARY} \rangle \mid \langle \text{UNARY OPERATOR} \rangle \langle \text{PRIMARY} \rangle$

$\langle \text{PRIMARY} \rangle ::= \langle \text{EXPRESSION} \rangle$
 $\mid \langle \text{VARIABLE} \rangle$
 $\mid \langle \text{STRING} \rangle$
 $\mid \langle \text{CONSTANT IDENTIFIER} \rangle$
 $\mid \langle \text{INTEGER} \rangle$
 $\mid \langle \text{LOGICAL VALUE} \rangle$

$\langle \text{CONSTANT IDENTIFIER} \rangle ::= \text{(Refer to the CONSTANT Statement).}$

Semantics:

$\langle \text{EXPRESSION} \rangle$ s are used in various statements in the REQUEST and CONTROL Sections of the Network Definition Language. Note that no distinction is made between logical and arithmetic expressions. Expressions such as (TRUE + 10 NE DISCONNECT) * TALLY [0] are syntactically legal and will execute without run-time errors.

Operator Precedence:

Operator precedence determines the order of evaluation of an expression if there are no parentheses to guide evaluation, as in the expression: TRUE + 10 NE DISCONNECT.

Two rules of precedence are used:

- a. Operators of equal precedence are evaluated from left to right.
- b. Where operators have unequal precedence, they are evaluated from highest to lowest precedence. The precedence order appears below. Operators having equal precedence are listed on the same line.

UNARY +, UNARY –
*, (SLASH)
BINARY +, BINARY –
LS, LE, EQ, NE, GT, GE, =, (,)
NOT
AND
OR
XOR

Example:

The (EXPRESSION) : TRUE + 10 NE DISCONNECT * 3 is evaluated as if it had been written:

(TRUE + 10) NE (DISCONNECT * 3).

SYNTAX AND SEMANTICS OF SECTIONS

Syntax:

(NETWORK DESCRIPTION) ::= (DECLARATION SECTION)
(REQUEST SECTION)
(LINE CONTROL SECTION)
(TERMINAL SECTION)
(STATION SECTION)
(LINE SECTION)
(FILE SECTION)

Semantics:

A (NETWORK DESCRIPTION) is the physical and logical specification of a data communications network. The various parts of a (NETWORK DESCRIPTION) as listed above will be defined both syntactically and semantically in following sub-sections of this document. The definitions will appear in the order in which they must be presented to the NDL Compiler for processing.

Six of the seven sections that make up the (NETWORK DESCRIPTION) are required. The (DECLARATION SECTION) can be omitted.

The statements that are required or result in a warning message when omitted are found in Appendix C, List of Required Statements.

Declaration Section

Syntax:

```

DECLARATION SECTION ::= <DECLARATION> | <DECLARATION>
                        <DECLARATION SECTION>

DECLARATION ::= DECLARATION <IDENTIFIER> : <DECLARATION LIST>

DECLARATION LIST ::= <DECLARATION> . | <DECLARATION LIST>
                    <DECLARATION> . | <EMPTY> .

DECLARATION ::= <AUDITFILE DECLARATION>
               | <CONSTANT DECLARATION>
               | <DEFINE STATEMENT>
               | <MAX TALLY/TOG DECLARATION>
               | <MAX BUFFERS DECLARATION>
               | <MAX FILES DECLARATION>
               | <MAX MESSAGE DECLARATION>
               | <NIF NAME DECLARATION>

```

Semantics:

A DECLARATION section is optional. It provides a means of specifying global definitions for the user's Network Controller. A more specific description of this section's functions and capabilities are found in the semantics of the various declarations. All DECLARATION sections must appear before the first REQUEST.

AUDITFILE Declaration

Syntax:

```

<AUDITFILE DECLARATION> ::= AUDITFILE <AUDITFILE IDENTIFIER>
                           ( <FILE ATTRIBUTE LIST> ).

<FILE ATTRIBUTE LIST> ::= <FILE ATTRIBUTE>
                          | <FILE ATTRIBUTE> , <FILE ATTRIBUTE LIST>

<FILE ATTRIBUTE> ::= <LABEL PART>
                    | <DEVICE PART>
                    | <BUFFERS PART>
                    | <VARIABLE PART>
                    | <SAVE PART>
                    | <RECORDS PART>
                    | <AREAS PART>
                    | <EMPTY>

<LABEL PART> ::= LABEL = <FILE IDENTIFIER>

<FILE IDENTIFIER> ::= <FAMILY IDENTIFIER>
                    | <FAMILY IDENTIFIER> <SLASH> <FILE IDENTIFIER>
                    | <PACK IDENTIFIER> <SLASH> <FAMILY IDENTIFIER> <SLASH>
                    <FILE IDENTIFIER>

<FILE IDENTIFIER> ::= <EBCDIC STRING>

<FAMILY IDENTIFIER> ::= <EBCDIC STRING>

<PACK IDENTIFIER> ::= <EBCDIC STRING>

<DEVICE PART> ::= DEVICE = <DEVICE SPECIFIER>

```

**DECLARATION
SECTION**

⟨ DEVICE SPECIFIER ⟩ ::= DISK (Any DISK)
| DISKFILE (Head-Per-Track)
| DISKPACK
| DISKCARTRIDGE
| TAPE (Any TAPE)
| TAPE 7 (7-Track)
| TAPE 9 (9-Track)
| TAPEPE (Phase-Encoded)
| PRINTER
| PRINTER or BACKUP

⟨ BUFFERS PART ⟩ ::= BUFFERS = ⟨ INTEGER ⟩

⟨ VARIABLE PART ⟩ ::= VARIABLE

NOTE

This indicates variable-length records.

⟨ SAVE PART ⟩ ::= SAVE = ⟨ INTEGER ⟩

NOTE

This is the number of days the file is to be saved.

⟨ RECORDS PART ⟩ ::= RECORDS = ⟨ RECORD SPECIFIER ⟩

⟨ RECORD SPECIFIER ⟩ ::= ⟨ RECORD LENGTH ⟩
| ⟨ RECORD LENGTH ⟩ ⟨ SLASH ⟩ ⟨ BLOCK LENGTH ⟩

⟨ RECORD LENGTH ⟩ ::= ⟨ INTEGER ⟩

⟨ BLOCK LENGTH ⟩ ::= ⟨ INTEGER ⟩

NOTE

All lengths are specified in bytes.

⟨ AREAS PART ⟩ ::= AREAS = ⟨ NUMBER AREAS ⟩ ⟨ SLASH ⟩ ⟨ BLOCKS PER AREA ⟩

⟨ NUMBER AREAS ⟩ ::= ⟨ INTEGER ⟩

⟨ BLOCKS PER AREA ⟩ ::= ⟨ INTEGER ⟩

NOTE

The AREAS attribute applies to disk files only.

Semantics:

Auditfiles are used in the `< AUDIT STATEMENT >` in the REQUEST Section, and any auditfile may be referenced in any REQUEST in the NDL program. A maximum of three auditfiles may be declared. Auditfiles are simple, sequential files that are opened OUTPUT by the user's Network Controller, as far as the MCP is concerned. This permits them to be modified at BOJ time by the FILE statement (refer to B 1800/B 1700 SYSTEM SOFTWARE OPERATIONAL GUIDE, form number 1068731).

The internal file name is the `< AUDITFILE IDENTIFIER >`. The default file attributes are:

DEVICE = DISK, LABEL = `< AUDITFILE IDENTIFIER >`, BUFFERS = 1, Fixed length RECORDS, SAVE = 30.

If DEVICE = DISK then RECORDS = 180 (unblocked) and AREAS = 40 (100 Records Per Area).

If DEVICE = TAPE then RECORDS = 180 (unblocked).

If DEVICE = PRINTER then RECORDS = 132 (unblocked), and the file will be written to backup disk or tape. The LOCK bit in the File Parameter Block (FPB) is set to insure CLOSE WITH LOCK if the Network Controller is "DS-ed".

Examples:

1. AUDITFILE AUDITDISK (DEVICE = DISKCARTRIDGE, LABEL = "PACKNAME"/"MYFAMILY"/"MYFILE", BUFFERS = 2, VARIABLE, SAVE = 7, RECORDS = 180/10, AREAS = 1/1000).
2. AUDITFILE DEFAULT
3. AUDITFILE AUDITTAPE (DEVICE = TAPE, LABEL = "AUDITFILE"/"A", RECORDS = 180/1).
4. AUDITFILE PRNTR (DEVICE = PRINTER).

CONSTANT Declaration

Syntax:

`< CONSTANT DECLARATION > ::= CONSTANT < CONSTANT DEFINITION LIST >`

`< CONSTANT DEFINITION LIST > ::= < CONSTANT DEFINITION >
| < CONSTANT DEFINITION >, < CONSTANT DEFINITION LIST >`

**DECLARATION
SECTION**

⟨ CONSTANT DEFINITION ⟩ ::= ⟨ CONSTANT IDENTIFIER ⟩ = ⟨ STRING ⟩

⟨ CONSTANT IDENTIFIER ⟩ ::= ⟨ IDENTIFIER ⟩

Semantics:

The CONSTANT declaration provides the user with a way to equate identifiers with strings of characters and control characters for which no EBCDIC graphic exists. In subsequent references, occurrence of the identifier causes the compiler to substitute the equated string. This is useful for frequently used strings such as control characters, or common concatenations of control characters.

Twenty-eight common control characters have been defined within the compiler to equal strings, as shown in Table 3-2. These identifiers may be redefined to equal other strings, but may not be used as non-constant identifiers.

Table 3-2.

NDL Compiler-Defined Identifiers

Identifier	String	Identifier	String
ACK	= 4"2E"	FSL	= 4"A2"
BEL	= 4"2F"	GS	= 4"1D"
CAN	= 4"18"	HT	= 4"05"
CR	= 4"0D"	LF	= 4"25"
DC1	= 4"11"	NAK	= 4"3D"
DC2	= 4"12"	NUL	= 4"00"
DC3	= 4"13"	POL	= 4"97"
DC4	= 4"3C"	RS	= 4"1E"
DEL	= 4"07"	SI	= 4"0F"
DLE	= 4"10"	SEL	= 4"98"
ENQ	= 4"2D"	SO	= 4"0E"
EOT	= 4"37"	SOH	= 4"01"
ESC	= 4"27"	STX	= 4"02"
ETB	= 4"26"	SYN	= 4"32"
ETX	= 4"03"	US	= 4"1F"
FF	= 4"0C"	VT	= 4"0B"
FS	= 4"1C"		

Examples of Constants:

```
CONSTANT NUL = 4"00"
          BAD1 = "ERROR",
          RETRANSMIT = "PLEASE SEND AGAIN".
```

```
CONSTANT QUOTE = " " " " " "
```

```
CONSTANT CRLF = 4"0D250000".
```

The following examples are invalid:

```
CONSTANT 12KANGAROO =           % The identifier must begin with
                                % a letter.
12. % This is not a string.
```

CONSTANT CLEAR = % A reserved word may not be
 % used as a constant.
 4"000" % This is a malformed string.

DEFINE Statement

Syntax:

⟨ DEFINE STATEMENT ⟩ ::= DEFINE ⟨ DEFINE DECLARATION LIST ⟩.

⟨ DEFINE DECLARATION LIST ⟩ ::= ⟨ DEFINE DECLARATION ⟩ |
 ⟨ DEFINE DECLARATION ⟩ ,
 ⟨ DEFINE DECLARATION LIST ⟩

⟨ DEFINE DECLARATION ⟩ ::= ⟨ DEFINE ID ⟩ = # ⟨ DEFINITION ⟩ #

⟨ DEFINE ID ⟩ ::= ⟨ IDENTIFIER ⟩

⟨ DEFINITION ⟩ ::= ⟨ TEXT ⟩

⟨ TEXT ⟩ ::= Any sequence of characters except a “#” which is not contained
 within a character string.

Semantics:

Beginning with the first REQUEST section, any ⟨ DEFINE ID ⟩ found by the compiler is replaced by its ⟨ DEFINITION ⟩. Nested DEFINE statements are permitted with no restriction on the number of levels.

Restrictions:

- a. ⟨ DEFINE ID ⟩ may not be an NDL reserved word, or an ⟨ IDENTIFIER ⟩ which is already a constant. ⟨ DEFINE ID ⟩ cannot be redefined.
- b. CONTROL (“\$”) cards must not contain defined symbols.
- c. Any CONTROL card detected while processing a DEFINE statement is ignored.
- d. Comments (“%”) are allowed within the ⟨ DEFINITION ⟩, but are discarded during processing. Also, all excess blanks are removed.
- e. A ⟨ DEFINE ID ⟩ must not define itself.
- f. A ⟨ DEFINITION ⟩ must not exceed 327 characters.
- g. Parametric defines are not allowed.
- h. Nested defines are expanded without regard to the number of levels until the total size of the expansion and remaining source image exceeds approximately 400 characters.
- i. All DEFINE statements are global.

DECLARATION SECTION

Examples:

```
DEFINE CHECKRETRY = # IF RETRY GO O
```

```
THEN DO.
```

```
    RETRY := RETRY - 1.  
    TERMINATE NOINPUT.
```

```
END. #
```

```
DEFINE REQUESTHRU = # LINE (CONTROL KEY)
```

```
    = 0 #
```

```
DEFINE JINX = # RETRY = 0 #
```

MAX TALLY/TOG Declaration

Syntax:

```
< MAX TALLY/TOG DECLARATION > ::= MAX TALLY [ < INTEGER > ].  
    | MAX LINE (TALLY [ < INTEGER > ] ).  
    | MAX TOG [ < INTEGER > ].  
    | MAX LINE (TOG [ < INTEGER > ] ).
```

Semantics:

Variable indexes can be used with TALLIES and TOGGLES in the REQUEST and CONTROL Sections of the NDL program. When variable indexes are used, a corresponding MAX TALLY/TOG statement must be included in the DECLARATION Section, and must explicitly state the largest TALLY or TOGGLE to be accessed.

The < INTEGER > specifies the largest allowable TALLY or TOGGLE and therefore limits the TALLY or TOGGLE section of the LINE or STATION table.

A MAX TALLY/TOG statement is not required if variable indexes are not used, and maximum values are determined by the largest index used.

Restrictions:

- a. No more than 100 TALLIES or TOGGLES may be declared, making the largest permissible index 99.
- b. If the declared maximum is exceeded by an explicit TALLY or TOGGLE index, a warning results and the maximum is increased to include the desired index.

Example:

DECLARATION:

MAX LINE (TOG[2]).
MAX TOG [99].

REQUEST R1:

```
.
.
.
IF TOG [LINE] THEN . . .
.
.
IF LINE (TOG[LINE(TALLY[1])]) THEN . . .
```

MAX BUFFERS DECLARATION

Syntax:

⟨ MAX BUFFERS DECLARATION ⟩ ::= MAX BUFFERS = [(INTEGER)].

Semantics:

The MAX BUFFERS declaration allows the user to declare the maximum number of buffers, up to 255, in the station queue (output) part of a remote file which will remain in memory. The default is 2.

MAX FILES DECLARATION

Syntax:

⟨ MAX FILES DECLARATION ⟩ ::= MAX FILES = [(INTEGER)].

Semantics:

The MAX FILES declaration allows the user to increase the number of application programs opening a remote file, up to 255, to a number greater than the number of remote files declared in the file section of NDL. For example: if one program opens a remote file INPUT-OUTPUT and another program opens a file OUTPUT only, MAX FILES must be increased to one over what it would default to in order to accommodate the second open. The default value of MAX FILES is the number of files declared in the file section of NDL.

MAX MESSAGES DECLARATION

Syntax:

⟨ MAX MESSAGES DECLARATION ⟩ ::= MAX MESSAGES = [(INTEGER)].

Semantics:

The MAX MESSAGES declaration allows the user to declare the maximum number of buffers, up to 255, that will be allocated for the station queue (output) part of a remote file. The difference between MAX MESSAGES and MAX BUFFERS is the number of buffers in a station queue (output) part of a remote file which will be written to disk. The default is 20.

DECLARATION SECTION

NIF NAME Declaration

Syntax:

```
< NIF NAME DECLARATION > ::= NIF = “< FILE IDENTIFIER >”.  
    | “< FILE IDENTIFIER >” < SLASH > “< FILE IDENTIFIER > ”.  
    | “< CARTRIDGE IDENTIFIER >” < SLASH >  
        “< FILE IDENTIFIER >” < SLASH >.  
    | “< CARTRIDGE IDENTIFIER >” < SLASH >  
        “< FILE IDENTIFIER >” < SLASH > “< FILE IDENTIFIER >”.
```

Semantics:

The NIF NAME Declaration allows the user the option of changing the name of the NIF file without using file equate cards. The external file name in the NIF file created by the NDL Compiler and the compiled Network Controller is changed to < FILE IDENTIFIER >.

The < FILE IDENTIFIER > must be enclosed in quote marks (“”) and can include special characters. If the < FILE IDENTIFIER > exceeds ten characters it will be truncated.

The < CARTRIDGE IDENTIFIER > is the name that is assigned to a user cartridge or user disk pack at initialization time.

When the NIF declaration is not used, the default name “NDL/NIF” is used.

Examples:

```
NIF = “ON*SYSTEM”/“DISK&-%$”.
```

```
NIF = “NDL”/“&THIS”/.
```

Request Section**Syntax:**

```

< REQUEST SECTION > ::= < REQUEST > | < REQUEST > < REQUEST SECTION >

< REQUEST > ::= REQUEST < IDENTIFIER > : < REQUEST STATEMENT LIST >

< REQUEST STATEMENT LIST > ::= < REQUEST STATEMENT > .
                               | < REQUEST STATEMENT > . < REQUEST STATEMENT LIST >

< REQUEST STATEMENT > ::= < ASSIGNMENT STATEMENT >
                          | < ATTACH OUTPUT STATEMENT >
                          | < AUDIT STATEMENT >
                          | < CASE STATEMENT >
                          | < DECREMENT STATEMENT >
                          | < DETACH OUTPUT STATEMENT >
                          | < DISPLAY STATEMENT >
                          | < DO STATEMENT >
                          | < DUMP STATEMENT >
                          | < FETCH STATEMENT >
                          | < FINISH STATEMENT >
                          | < IF STATEMENT >
                          | < INCREMENT STATEMENT >
                          | < INITIALIZE STATEMENT >
                          | < INITIATE STATEMENT >
                          | < NULL STATEMENT >
                          | < QUEUE INPUT STATEMENT >
                          | < QUEUE OUTPUT STATEMENT >
                          | < RECEIVE STATEMENT >
                          | < TERMINATE STATEMENT >
                          | < TRANSMIT STATEMENT >
                          | < UNDO STATEMENT >

```

Semantics:

The REQUEST Section consists of one or more REQUESTS, which are line discipline routines that are used by the Network Controller in communicating with the remote devices of the Data Comm network. A REQUEST must be defined for each capability of a terminal; if it is possible for a terminal to send input to the system and to receive output from the system, then both a receive and a transmit REQUEST must be defined for that terminal. The REQUEST to be used for each of these capabilities is specified by the < TERMINAL REQUEST STATEMENT >, which is described in the TERMINAL Section of this document.

The Network Controller initiates these REQUESTS upon command from the various parts of the Data Comm system. For instance, if it is necessary to output a message to a TC500 the Network Controller selects the correct transmit REQUEST as specified in the < TERMINAL REQUEST STATEMENT >, and executes the statements within that REQUEST. REQUESTS are re-entrant, so that two stations using the same REQUEST may be handled concurrently on different lines.

REQUEST SECTION

REQUESTS are treated as I/O co-routines by the Network Controller. When a LINE CONTROL needs to send a message to a TC500, the Network Controller selects the correct transmit REQUEST from the TC500 terminal definition and gives it control. The TC500 transmit REQUEST executes the line discipline until the I/O is successful or until it detects a serious error, at which point the Network Controller regains control. The REQUEST will also yield control to the Network Controller while hardware I/O is in progress for the TC500 station. Until the I/O is complete, the Network Controller can process other lines or yield control to the MCP. This allows several Data Comm I/O processes to be handled concurrently by the Network Controller, and is transparent to the REQUEST. REQUESTS are coded as if I/O were instantaneous.

A file of library REQUESTS exists to handle common line disciplines. Refer to appendix D.

There are four statements that can be used in the REQUEST Section that are used for transfer of control. They are the DO, UNDO, CASE, and IF statements. There is no way to declare subroutines for a REQUEST, and local variables are restricted to those defined in the section titled Basic Components.

The syntax and semantics of the 16 statements that can be used in REQUESTS follow.

ASSIGNMENT Statement

Syntax:

⟨ ASSIGNMENT STATEMENT ⟩ ::= ⟨ ASSIGNABLE VARIABLE ⟩ := ⟨ EXPRESSION ⟩ .

⟨ ASSIGNABLE VARIABLE ⟩ ::= (See Basic Components).

Semantics:

The ⟨ ASSIGNMENT STATEMENT ⟩ is used in REQUESTS and CONTROLS to change the value of an assignable VARIABLE.

Examples:

```
TALLY [0] := TALLY [1] * 3.  
TOG [1] := LINE (TOG [0]).  
TIME (TALLY) := TIME.
```

ATTACH OUTPUT STATEMENT

Syntax:

⟨ ATTACH OUTPUT STATEMENT ⟩ ::= ATTACH OUTPUT

Semantics:

This statement takes the first message from a station queue and places it in the appropriate line buffer. This statement is the exact opposite of QUEUE OUTPUT.

AUDIT Statement

Syntax:

⟨ AUDIT STATEMENT ⟩ ::= AUDIT ⟨ AUDITFILE IDENTIFIER ⟩ (⟨ AUDIT ELEMENT LIST ⟩)

⟨ AUDIT IDENTIFIER ⟩ ::= (See AUDITFILE Declaration in the DECLARATION Section)

< AUDIT ELEMENT LIST > ::= < MESSAGE > | < EXPRESSION LIST >
 | < EXPRESSION LIST >, < MESSAGE >
 | < MESSAGE >, < EXPRESSION LIST >
 | < EXPRESSION LIST >, < MESSAGE >, < EXPRESSION LIST >

< MESSAGE > ::= BUFFER (< MESSAGE MODIFIER >)
 | TEXT (< MESSAGE MODIFIER >)

< MESSAGE MODIFIER > ::= < IO > | < IO >, < INTEGER >

< IO > ::= INPUT | OUTPUT

< EXPRESSION LIST > ::= < EXPRESSION > | < EXPRESSION >, < EXPRESSION LIST >

Semantics:

The < AUDIT STATEMENT > causes a write to a user-declared file. The data to be written is specified by the < AUDIT ELEMENT LIST >. As shown in the syntax, the < AUDIT ELEMENT LIST > can be formed by concatenating a combination of < MESSAGE > and < EXPRESSION LIST >. Blanks will be added to the least significant character positions if the concatenated string is shorter than the AUDITFILE RECORD LENGTH. If the concatenated string is longer than the AUDITFILE RECORD LENGTH, truncation of the least significant character positions will occur.

The elements of < MESSAGE > are further described as follows:

- a. TEXT is that part of the message excluding the header.
- b. BUFFER refers to the message including the 50-byte header.

REQUEST SECTION

The \langle INTEGER \rangle option of the \langle MESSAGE MODIFIER \rangle tells the Network Controller how many characters of the \langle MESSAGE \rangle to audit.

Example:

```
AUDIT FILEA (TEXT (INPUT, 10) ).
```

This statement results in the first 10 characters of the TEXT being written to FILEA.

\langle EXPRESSION LIST \rangle may be included before, after, or without \langle MESSAGE \rangle . The fields resulting from the evaluation of each expression are concatenated in the order specified. Refer to Expressions for the length of expression values. The String functions, DECIMAL and CONVERT are useful for changing TALLIES, TOGGLES, and FLAGS to a format that is easy to print.

Example:

```
AUDIT FILEA ( "TALLY1 = ", DECIMAL (TALLY [1], 3), " , INPUT = ",  
TEXT (INPUT, 10), "THATS ALL" ).
```

The above AUDIT statement results in a 36-character string consisting of: "TALLY1 = nnn, INPUT = ttttttttTHATS ALL".

CASE Statement

Syntax:

```
 $\langle$  CASE STATEMENT  $\rangle$  ::=  $\langle$  CASE HEAD  $\rangle$   
                                   $\langle$  CASE BODY  $\rangle$ 
```

```
 $\langle$  CASE HEAD  $\rangle$  ::= CASE  $\langle$  EXPRESSION  $\rangle$ .
```

```
 $\langle$  CASE BODY  $\rangle$  ::=  $\langle$  REQUEST STATEMENT LIST  $\rangle$   
                           $\langle$  CASE ENDING  $\rangle$ 
```

```
 $\langle$  CASE ENDING  $\rangle$  ::= END CASE
```

Semantics:

The \langle EXPRESSION \rangle serves as an index into the list of \langle REQUEST STATEMENT \rangle s. Only the selected statement is executed, and control then goes to the statement following the \langle CASE ENDING \rangle , unless the statement is an UNDO.

If there are n number of statements in the list, the range of < EXPRESSION > may be from 0 through n-1.

The statements in the list may be any legal < REQUEST STATEMENT > allowed in NDL. If the user wants to execute “nothing” in a given case, the < NULL STATEMENT > is appropriate.

DECREMENT Statement

Syntax:

< DECREMENT STATEMENT > ::= DECREMENT TRAN < TRAN MODIFIER >

< TRAN MODIFIER > ::= (TRANSMIT) | (RECEIVE)

Semantics:

The DECREMENT statement decreases the value of an eight-bit, decimal-encoded transmission number by one. It is intended to be used with Data Comm systems where the MCS does not maintain the output transmission numbers.

< TRAN MODIFIER > can be one of these:

- a. (TRANSMIT) - output transmission number.
- b. (RECEIVE) - input transmission number.

Example:

Assume that the TRANSMISSION NUMBER of the last output message was “380”.
DECREMENT TRAN (TRANSMIT) will decrease the value to “379” (hexadecimal F3F7F9).
The field will be set to all “9”s on underflow.

DETACH OUTPUT STATEMENT

Syntax:

< DETACH OUTPUT STATEMENT > ::= DETACH OUTPUT

Semantics:

This statement discards any output message attached to the line. The request does not terminate.

DISPLAY Statement

Syntax:

< DISPLAY STATEMENT > ::= DISPLAY < EXPRESSION LIST >

< EXPRESSION LIST > ::= < EXPRESSION > | < EXPRESSION > , < EXPRESSION LIST >

Semantics:

The DISPLAY statement allows a message to be printed on the console printer. < EXPRESSION > must be a literal, constant identifier, or variable name.

REQUEST SECTION

Example:

```
DISPLAY "LINE", DECIMAL(LINE,1), " , STATION " ,  
DECIMAL(STATION,1) " IS NOT RESPONDING".
```

When the DISPLAY statement shown in the example above is executed the following output would result on the console printer:

```
LINE 1, STATION 3 IS NOT RESPONDING
```

DO Statement

Syntax:

⟨ DO STATEMENT ⟩ ::= ⟨ DO HEAD ⟩ ⟨ REQUEST STATEMENT LIST ⟩ ⟨ END PART ⟩

⟨ DO HEAD ⟩ ::= DO ⟨ IDENTIFIER PART ⟩ ⟨ FOREVER PART ⟩

⟨ END PART ⟩ ::= END ⟨ IDENTIFIER PART ⟩.

⟨ IDENTIFIER PART ⟩ ::= ⟨ IDENTIFIER ⟩ | ⟨ EMPTY ⟩

⟨ FOREVER PART ⟩ ::= FOREVER | ⟨ EMPTY ⟩

Semantics:

The DO statement provides a means of grouping two or more REQUEST statements (including other DO statements) as one statement. This is useful in ⟨ IF STATEMENTS ⟩ and ⟨ CASE STATEMENTS ⟩. DO statements may be nested. The ⟨ IDENTIFIER PART ⟩ and the ⟨ FOREVER PART ⟩ are optional.

Example:

```

IF TOG [1] THEN
  DO OUTER.
    DO INNER FOREVER.
      FETCH CHARACTER.
      IF CHAR EQ ETX THEN
        UNDO INNER.
      END INNER.
      TALLY [1] := TALLY [1] + 1.
    END OUTER.

```

When the < FOREVER PART > is used it causes the < REQUEST STATEMENT LIST > to be repeatedly executed until an UNDO or a TERMINATE is executed.

Restrictions:

1. If a < DO IDENTIFIER > is included in the < DO HEAD >, it must also be included in the < END PART >.
2. If the < DO HEAD > does not include a < DO IDENTIFIER >, the < END PART > must not include one.

DUMP Statement

Syntax:

< DUMP STATEMENT > ::= DUMP

Semantics:

When DUMP is specified, the contents of a programs memory space is dumped to disk for subsequent analysis by the NDL/DUMP program. Processing automatically continues when the dump is finished.

FETCH Statement

Syntax:

< FETCH STATEMENT > ::= FETCH < POINTER ADVANCE PART > < FETCH OBJECT >

< POINTER ADVANCE PART > ::= 0 | 1 | < EMPTY >

< FETCH OBJECT > ::= CHAR | CHARACTER | < EMPTY >

Semantics:

The FETCH statement is used only in input REQUESTS. Before using FETCH an INITIATE RECEIVE should be executed. All < FETCH OBJECT > s have the same meaning; FETCH always loads the character register with the character in the input message that is currently addressed by the message pointer. The message pointer is advanced by one character after the FETCH if the < POINTER ADVANCE PART > is 1 or < EMPTY >. A 0 in the < POINTER ADVANCE PART > leaves the message pointer unchanged, so a subsequent FETCH or RECEIVE accesses the same character as before.

NOTE

ENDOFBUFFER will be set if the FETCH is executed with the message pointer set beyond the last character in the buffer.

REQUEST SECTION

FINISH Statement

Syntax:

⟨ FINISH STATEMENT ⟩ ::= FINISH TRANSMIT ⟨ TRANSMIT PART ⟩ ⟨ TRANSMIT-RECEIVE PART ⟩

⟨ TRANSMIT PART ⟩ ::= (⟨ TRANSMIT LIST ⟩) | ⟨ EMPTY ⟩

⟨ TRANSMIT LIST ⟩ ::= ⟨ TRANSMIT ATTRIBUTE ⟩
 ⟨ TRANSMIT ATTRIBUTE ⟩ , ⟨ TRANSMIT LIST ⟩

⟨ TRANSMIT ATTRIBUTE ⟩ ::= ⟨ TRANSMIT BASIC ATTRIBUTE ⟩
 | NO ⟨ TRANSMIT BASIC ATTRIBUTE ⟩

⟨ TRANSMIT BASIC ATTRIBUTE ⟩ ::= EOT | CR | TRANSLATE | ⟨ EMPTY ⟩

⟨ TRANSMIT-RECEIVE PART ⟩ ::= INITIATE RECEIVE ⟨ RECEIVE PART ⟩

⟨ RECEIVE PART ⟩ (Refer to the INITIATE statement)

Semantics:

The FINISH TRANSMIT statement is required to cause transmission of output message that has been constructed by a series of ⟨ TRANSMIT STATEMENT ⟩s. A FINISH TRANSMIT-INITIATE RECEIVE will cause a line status change from TRANSMIT to RECEIVE if there were no errors on the transmit portion of the operation.

The ⟨ TRANSMIT BASIC ATTRIBUTE ⟩ sets variants in the hardware I/O descriptor; they are instructions to the B 1800/B 1700 Data Communications hardware, and have meanings as follows:

EOT is to be treated as a control character.

CR is to be treated as a control character (TELETYPE adapter only).

TRANSLATE is for EBCDIC translation to ASCII.

The defaults are: TRANSLATE,EOT,NO CR.

IF Statement

Syntax:

⟨ IF STATEMENT ⟩ ::= IF ⟨ EXPRESSION ⟩ THEN ⟨ REQUEST STATEMENT ⟩
 ⟨ ELSE PART ⟩

⟨ ELSE PART ⟩ ::= ELSE ⟨ REQUEST STATEMENT ⟩ | ⟨ EMPTY ⟩

Semantics:

The ⟨ EXPRESSION ⟩ is evaluated, and if the least significant bit of the value is equal to one (TRUE), the statement following THEN will be executed. If the bit is off (FALSE) the statement following ELSE (if present) will be executed. Control always passes to the statement following the IF STATEMENT. Any ⟨ REQUEST STATEMENT ⟩ is legal after THEN and ELSE including ⟨ DO STATEMENT ⟩ s, ⟨ IF STATEMENT ⟩ s, and ⟨ NULL STATEMENT ⟩ s.

Examples:

1. IF CHAR NE ACK THEN
DO.
RECEIVE ADDRESS.
IF ADDERR THEN
TERMINATE ERROR.
END.
ELSE
INITIATE RECEIVE.
2. IF CHAR = "R" THEN
TALLY [0] := 1.
ELSE
TALLY [0] := 2.

INCREMENT Statement

Syntax:

⟨ INCREMENT STATEMENT ⟩ ::= INCREMENT TRAN ⟨ TRAN MODIFIER ⟩

⟨ TRAN MODIFIER ⟩ ::= (TRANSMIT) | (RECEIVE)

Semantics:

The INCREMENT statement increases the value of an eight-bit, decimal-encoded transmission number by one. It is intended to be used with Data Communications systems where the MCS does not maintain the output transmission numbers.

⟨ TRAN MODIFIER ⟩ can be one of these:

- a. TRANSMIT - output transmission number.
- b. RECEIVE - input transmission number.

Example:

Assume that the output TRANSMISSION NUMBER of the last message was "379". INCREMENT TRAN (TRANSMIT) will increment the value to "380" (hexadecimal F3F8F0). The field will be set to character zeroes on overflow.

INITIALIZE Statement

Syntax:

⟨ INITIALIZE STATEMENT ⟩ ::= INITIALIZE ⟨ ITEM LIST ⟩

⟨ ITEM LIST ⟩ ::= ⟨ ITEM ⟩ | ⟨ ITEM ⟩ , ⟨ ITEM LIST ⟩

⟨ ITEM ⟩ ::= TEXT | TRAN (RECEIVE) | TRAN (TRANSMIT)
| TALLY [⟨ TALLY INDEX ⟩] | TOG [⟨ TOG INDEX ⟩] | RETRY

REQUEST SECTION

Semantics:

INITIALIZE TEXT sets the input message pointer in the Network Controller to the first byte of the input buffer, including the terminal header. An INITIATE RECEIVE and the successful completion of AUTO-POLL does this automatically.

INITIALIZE TRAN (RECEIVE) resets the station input transmission number to binary zeroes.

INITIALIZE TRAN (TRANSMIT) sets the station output transmission number from the output external message header if a message is attached; otherwise, it is set to character zeroes.

INITIALIZE TALLY sets the listed TALLY(S) from the internal 50-byte message header if there is one, else to zero. Only TALLY [0-7] may be initialized from the 50-byte message header.

INITIALIZE TOGGLE sets the listed TOGGLE(S) from the internal 50-byte message header if there is one, else to zero. Only TOG [0-7] can be initialized from the message header.

INITIALIZE RETRY resets the retry to its starting value (refer to the RETRY statement in the STATION section of this document).

INITIATE Statement

Syntax:

⟨ INITIATE STATEMENT ⟩ ::= INITIATE ⟨ INITIATE ACTION ⟩

⟨ INITIATE ACTION ⟩ ::= RECEIVE ⟨ RECEIVE PART ⟩ | TRANSMIT

⟨ RECEIVE PART ⟩ ::= ⟨ EMPTY ⟩ | (⟨ RECEIVE LIST ⟩)

⟨ RECEIVE LIST ⟩ ::= ⟨ RECEIVE ATTRIBUTE ⟩
| ⟨ RECEIVE ATTRIBUTE ⟩ , ⟨ RECEIVE LIST ⟩

⟨ RECEIVE ATTRIBUTE ⟩ ::= ⟨ RECEIVE BASIC ATTRIBUTE ⟩
| NO ⟨ RECEIVE BASIC ATTRIBUTE ⟩

⟨ RECEIVE BASIC ATTRIBUTE ⟩ ::= TRANSLATE | TIMEOUT
| EOT | CR | PARITY | ⟨ EMPTY ⟩

Semantics:

The INITIATE RECEIVE form of the INITIATE STATEMENT commands the adapter (i.e., line) to accept input.

The INITIATE TRANSMIT sets the output message pointer in the Network Controller to the first byte in the output message buffer.

The TRANSLATE attribute causes hardware translation of data received.

TIMEOUT enables the receive timeout timer.

EOT causes the adapter to treat EOT as a control character.

CR causes the adapter to treat CR as a control character (TELETYPE adapter only).

PARITY causes parity checking of each character (TELETYPE adapter only). Other adapters parity checking will not be affected by this option.

The defaults are: TRANSLATE
TIMEOUT
EOT
NO CR
NO PARITY

NULL Statement

Syntax:

⟨ NULL STATEMENT ⟩ ::= ⟨ EMPTY ⟩

Semantics:

The NULL statement may be used in an IF statement or a CASE statement where an executable statement is required but a “no-operation” is desired.

Example:

```

CASE TALLY [0].
  IF TOG [0] THEN           %           CASE entry 0.
                           % NULL.
  ELSE
    INITIATE INPUT.
  DO.                       % NULL.    CASE entry 1.
                           %           CASE entry 2.
    ⟨ REQUEST STATEMENT LIST ⟩
  END.
END CASE.

```

In the example, the CASE contains three executable statements, an IF, a NULL, and a DO. One of the three will be executed depending on the value of TALLY [0]. The possibilities are:

- a. If TALLY [0] is equal to zero, the IF statement will be executed. If the THEN branch is taken, a NULL results and the CASE statement is exited.
- b. If TALLY [0] is equal to one the NULL statement is executed and the CASE is exited.
- c. If TALLY [0] is equal to two, the DO statement is executed.

QUEUE INPUT STATEMENT

Syntax:

⟨ QUEUE INPUT STATEMENT ⟩ ::= QUEUE INPUT

Semantics:

This statement causes a message to be sent to an application program after the execution of RECEIVE TEXT. The request does not terminate.

QUEUE OUTPUT STATEMENT

Syntax:

⟨ QUEUE OUTPUT STATEMENT ⟩ ::= QUEUE OUTPUT

REQUEST SECTION

Semantics:

This statement puts a message which is currently in a line buffer back at the top of the appropriate station queue. The request will not terminate. This statement is the exact opposite of ATTACH OUTPUT.

RECEIVE Statement

Syntax:

⟨ RECEIVE STATEMENT ⟩ ::= RECEIVE ⟨ ITEM LIST ⟩

⟨ ITEM LIST ⟩ ::= ⟨ ITEM ⟩ | ⟨ ITEM ⟩ , ⟨ ITEM LIST ⟩

⟨ ITEM ⟩ ::= ADDRESS ⟨ ADDRESS QUALIFIER ⟩ | TEXT | ⟨ STRING ⟩
| ⟨ CONSTANT ⟩ | TRAN ⟨ TRAN QUALIFIER ⟩

⟨ ADDRESS QUALIFIER ⟩ ::= ⟨ EMPTY ⟩ | (RECEIVE) | (TRANSMIT) | (STATION)

⟨ TRAN QUALIFIER ⟩ ::= (RECEIVE) | (TRANSMIT) | ⟨ empty ⟩

Semantics:

The ⟨ RECEIVE STATEMENT ⟩ provides the means to test characters, and is used in input REQUESTS. A ⟨ RECEIVE STATEMENT ⟩ must be preceded by an INITIATE RECEIVE. Starting with the current setting of the input message pointer, each item in the ⟨ ITEM LIST ⟩ is tested against the corresponding field in the input message. If they do not agree, the appropriate error flag is set.

⟨ ITEM ⟩ s are described as follows:

ADDRESS ⟨ EMPTY ⟩ defaults to ADDRESS (RECEIVE).

ADDRESS (RECEIVE) and ADDRESS (TRANSMIT) refer to the strings (number of address characters) specified in the station's ⟨ STATION ADDRESS STATEMENT ⟩. See the STATION Section of this document. If the proper number of characters in the input message do not agree with the address, ADDERR will be set.

ADDRESS (STATION). The address field in the message will be compared against the receive addresses of all stations on the current line. ADDERR will be set if no match is found. If a match is found, the station index is reset to that of the matching station.

NOTE

This has serious implications for REQUEST coding because all of the variables that are station-specific are now different. For instance, TALLY [0 - 19] and TOG [0 - 19] reference entirely different fields after a successful RECEIVE ADDRESS (STATION).

ADDRESS (STATION) provides the programmer with a means of implementing group poll, contention, and Remote Job Entry (RJE) line disciplines, since this is the only NDL construct that permits a REQUEST to change station index at run-time and respond to any of a set of stations which may have originated the current message.

TEXT. This attribute results in the input message pointer being set to the last character in the message with the assumption that a terminating control character follows. If more characters were received than were specified in the station's ⟨ BUFFERSIZE STATEMENT ⟩, ENDOFBUFFER will be set.

⟨ STRING ⟩. If the corresponding characters do not agree then FORMATERR will be set.

⟨ CONSTANT ⟩. FORMATERR will be set if the corresponding characters do not agree.

TRAN ⟨ empty ⟩ defaults to TRAN (RECEIVE).

TRAN (RECEIVE) refers to the transmission number characters of the input message. TRANERR will be set if the characters in the input message are equal to the referenced last transmission number.

TRAN (TRANSMIT) refers to the transmission number of the output message.

Example:

The following code appears within ⟨ DO STATEMENT ⟩ POLLTCD in REQUEST POLLTCTD in the sample NDL program in Appendix D of this manual.

```

FETCH CHARACTER.
IF CHARACTER EQ SOH THEN
  DO SOH1.
  RECEIVE ADDRESS.
  IF ADDERR THEN UNDO POLLTC5.
  RECEIVE TRAN.
  IF TRANERR THEN TOG [1] := 1.
  RECEIVE STX.
  IF FORMATERR THEN UNDO POLLTC5.
  RECEIVE TEXT.
  RECEIVE ETX.
  IF FORMATERR THEN UNDO POLLTC5.
END SOH1.

```

TERMINATE Statement

Syntax:

⟨ TERMINATE STATEMENT ⟩ ::= TERMINATE ⟨ TERMINATE OPTION ⟩

⟨ TERMINATE OPTION ⟩ ::= DISCONNECT
| ECHO
| ERROR
| INPUT
| INPUT (RETURN)
| INPUT (RETURN, NO BUFFER)
| LOGICALACK
| NOINPUT
| OUTPUT
| OUTPUT (RETURN)
Semantics: | RELEASE STATION

The TERMINATE STATEMENT is used to exit a REQUEST.

TERMINATE may cause:

- a. Buffer allocation and de-allocation.
- b. Message queuing and de-queuing.

REQUEST SECTION

- c. Changes in station status.
- d. Saving of “state” of the current REQUEST (see the Table 3-3, Terminate Options at the end of this sub-section. If “state” is saved when using the INPUT (RETURN) and OUTPUT (RETURN) options, the REQUEST will be restarted at the statement following the TERMINATE. If “state” is not saved the TERMINATE ends the REQUEST.

All REQUESTS must end with a “non-state saving” TERMINATE. No syntax error will result if such a TERMINATE is not supplied, but if control goes beyond the last statement in the REQUEST, the NDL Compiler will generate a “Terminate Error”.

The (TERMINATE OPTIONS) are further described as follows:

DISCONNECT is the terminate option that must be used when the last I/O operation resulted in an error indicating a loss of connection, such as LOSS OF DATA SET READY. A BREAK I/O with the disconnect variant set is issued. If stations are still ENABLED on the line, a new TEST/WAIT FOR RING is issued.

ERROR serves to inform the Network Controller of an unsuccessful attempt to transmit or receive. An error message will be queued for the MCS or the application program. Attached input buffers will be released and attached output messages will be re-queued.

INPUT is used in receive REQUESTS to signal completion of an input line discipline. The result index is set to indicate successful input of the message. The 50-byte message header is appended to the input text, and the input message is queued for the MCS/application program. The output buffer is de-allocated, since any output message still attached is assumed to have been successfully transmitted.

INPUT (RETURN) is used to signal successful reception of a block (not the last) of input. The message is received and queued, as in TERMINATE INPUT, and another input buffer is allocated. No action is taken on attached output messages. Control is returned to the next REQUEST statement in sequence so that it can begin reception of the next block of input.

INPUT (RETURN, NO BUFFER) signals successful reception on an input message, but not completion of the line discipline. The input message is received and queued, but a new input buffer is not allocated. No action is taken on output messages. Control is returned to the next REQUEST statement in sequence. This option is intended to be used to overlap processing of an input message by the MCS/application program with acknowledgement of reception by the REQUEST.

LOGICALACK is used to interrupt an input REQUEST. If an MCS is present it may record the message text, process it, and queue a reply before logically acknowledging the message. Several requirements that must be satisfied to use LOGICALACK are listed below:

- a. The optional LOGICALACK statement must be included in the STATION DEFINITION or the STATION DEFAULT DEFINITION for the affected station in the STATION Section of the NDL program.
- b. In the input REQUEST to be interrupted, a TERMINATE LOGICALACK statement must follow a successful RECEIVE TEXT operation.
 - 1. If LOGICALACK is “TRUE” for this station and an MCS is present, the input message will be queued for the MCS. The MCS must reply to the input message with a LOGICAL ACK-REPLY before the REQUEST will continue.

2. If LOGICALACK is “FALSE” or an MCS is not present, the input message will be queued for the application program and the REQUEST will continue (as with TERMINATE INPUT (RETURN, NO BUFFER)).

The Network Controller restarts the REQUEST unless an error is detected, in which case the error field indicates the following conditions:

- a. An invalid Logical Station Number in the LOGICALACK response.
- b. The line on which this LSN is located is not waiting for LOGICALACK.
- c. The value of STATION variable for this line does not correspond with the LOGICALACK reply LSN.

NOINPUT signals an unsuccessful attempt to receive input from a station, but without errors (e.g., receiving an EOT, ending a line discipline). The input buffer is de-allocated, and any output message is re-queued for an output REQUEST. “State” is lost. This option may be used in a transmit REQUEST for the purpose of re-queuing the output message, perhaps to broadcast a message to more than one station.

OUTPUT is used to indicate the successful completion of an output line discipline. A completion message is queued for the MCS if the MCS is present and participating in I/O, and if the MCS has requested the message. The output buffer and any attached input buffer are de-allocated, and the REQUEST ends.

OUTPUT (RETURN) signals the successful transmission of a block of output, but not the completion of the line discipline. The old output buffer is de-allocated, but no action is taken on any attached input buffer. The “state” of the REQUEST is saved, and the LINE CONTROL is entered (see the LINE (CONTROL KEY) = 4 option of the System Status Variables). The CONTROL must restart the REQUEST with a CONTINUE statement, which returns control to the next statement in sequence in the REQUEST. The REQUEST should test the value of OUTPUTATTACHED to determine if another output message has been attached before trying to transmit.

RELEASE STATION is used to release stations in a switched-line Data Comm environment. It clears the STATION table entry STATION (LINE) to zero. All stations of a line group that can dial in must be specified in the LINE STATION statement of the LINE Section. Since all of the stations specified need not be located together, stations not physically present on a given line can be released by TERMINATE RELEASE STATION, and thereby are available to other lines of the line group.

TRANSMIT Statement

Syntax:

```

< TRANSMIT STATEMENT > ::= TRANSMIT < ITEM LIST >

< ITEM LIST > ::= < ITEM > | < ITEM >, < ITEM LIST >

< ITEM > ::= ADDRESS < ADDRESS QUALIFIER > | TEXT | < STRING >
           | < CONSTANT IDENTIFIER > | CHAR | CHARACTER | < EXPRESSION >
           | TRAN < COMPLEX TRAN QUALIFIER >

< ADDRESS QUALIFIER > ::= (RECEIVE) | (TRANSMIT) | < EMPTY >

< COMPLEX TRAN QUALIFIER > ::= (RECEIVE) | (TRANSMIT) | < EMPTY >

```

Semantics:

The < TRANSMIT STATEMENT > is used for output operations to a station.

REQUEST SECTION

⟨ ITEM ⟩ s are concatenated until the ⟨ ITEM LIST ⟩ is exhausted.

⟨ ITEM ⟩ s from subsequent ⟨ TRANSMIT STATEMENT ⟩ s are added to the concatenation until a FINISH TRANSMIT STATEMENT is executed, at which time the entire string is actually sent to the station.

⟨ ITEM ⟩ s are described below:

ADDRESS ⟨ EMPTY ⟩ is the address from the message being processed (i.e., ADDRESS (TRANSMIT)).

ADDRESS (RECEIVE) and ADDRESS (TRANSMIT) - in each case the physical address of the terminal is fetched from the STATION table.

CHAR/CHARACTER - a value must be placed in the character register prior to the TRANSMIT.

⟨ CONSTANT IDENTIFIER ⟩ is a predefined symbol in Table 3-2, NDL Compiler — Defined Identifiers, in the DECLARATION Section or a default constant in NDL.

⟨ EXPRESSION ⟩. When the length of an expression is neither evenly divisible by eight, or a multiple of eight, a run error results. TRANSMIT TALLY (0) is a legal expression, because tallies are eight-bit fields, whereas TRANSMIT TOG (0), a one-bit field, causes a run error.

⟨ STRING ⟩ s may be transmitted.

TRAN (RECEIVE) and TRAN (TRANSMIT) - the transmission number of the terminal is fetched from the STATION table.

TRAN ⟨ empty ⟩ defaults to TRANSMIT TRAN (TRANSMIT).

In order to optimize the execution of REQUESTS, the coding techniques shown in the examples below should be used. Sequentially transmitted TRANSMIT ⟨ ITEM ⟩ s are concatenated into one string for efficiency, as illustrated by case A and case B of example 1.

Example 1:

```
Case A    TRANSMIT SOH, ADDRESS (TRANSMIT).  
          TRANSMIT TRAN (TRANSMIT).  
          TRANSMIT STX, TEXT, ETX.
```

The TRANSMIT instructions above are logically equivalent to the following TRANSMIT instructions.

```
Case B    TRANSMIT SOH, ADDRESS (TRANSMIT), TRAN (TRANSMIT), STX, TEXT, ETX.
```

Example 2:

```
Case C    TRANSMIT SOH.  
          IF TOG [ 3 ] THEN  
            TRANSMIT ADDRESS (TRANSMIT)  
          ELSE  
            TRANSMIT ADDRESS (RECEIVE)  
          TRANSMIT STX.
```

The TRANSMIT instructions in case C are less efficient than the TRANSMIT instructions in case D because the TRANSMIT $\langle \text{ITEM} \rangle$ s are separated by the IF/ELSE instructions in case C, and are therefore not sequential.

```

Case D      IF TOG [ 3 ] THEN
              TRANSMIT SOH, ADDRESS (TRANSMIT), STX.
            ELSE
              TRANSMIT SOH, ADDRESS (RECEIVE), STX.

```

In all cases, message text is not transmitted with the string of concatenated TRANSMIT $\langle \text{ITEM} \rangle$ s.

UNDO Statement

Syntax:

```

 $\langle \text{UNDO STATEMENT} \rangle ::= \text{UNDO } \langle \text{DO OPTION} \rangle$ 
 $\langle \text{DO OPTION} \rangle ::= \langle \text{EMPTY} \rangle \mid \langle \text{DO IDENTIFIER} \rangle$ 

```

Semantics:

The $\langle \text{UNDO STATEMENT} \rangle$ is used to exit a $\langle \text{DO STATEMENT} \rangle$. Execution of the UNDO will cause control to transfer to the statement following the $\langle \text{END PART} \rangle$ of the $\langle \text{DO STATEMENT} \rangle$.

The $\langle \text{DO OPTION} \rangle$ may take one of two forms:

- a. UNDO $\langle \text{empty} \rangle$ transfers control out of the innermost $\langle \text{DO STATEMENT} \rangle$ in which it appears.
- b. UNDO $\langle \text{DO IDENTIFIER} \rangle$ transfers control out of the DO statement with that $\langle \text{DO IDENTIFIER} \rangle$.

Example:

```

DO OUTER.
  DO.
    IF TOG [0] THEN UNDO.           % Exits the DO  $\langle \text{EMPTY} \rangle$ 
    DO INNER.
      IF TOG [1] THEN UNDO.       % Exits "INNER"
      UNDO OUTER.                 % Exits "OUTER"
    END INNER.
  END.
END OUTER.

```

Table 3-3. Terminate Options

TERMINATE OPTION	INPUT MESSAGE	OUTPUT MESSAGE	STATE	REQUEST TYPE	REMARKS
DISCONNECT	Lost	Lost	Lost	All	The line is disconnected and a wait for a ringing condition is initiated on that line. Makes all stations on that line available to all lines.
ECHO	Queued for Network Controller	NA	Lost	Debug	Places the input message in the N.C. output queue to allow station's output REQUEST to echo it back, rather than being queued for user program or MCS.
ERROR	Lost	Re-queued	Lost	Receive or Transmit	Informs N.C. of unsuccessful attempt to transmit or receive. Attached input buffers are released and attached output messages are re-queued.
INPUT	Queued	Lost	Lost	Receive	Signals the end of line discipline in receive REQUEST. The message header is appended to input text and input message is queued for MCS/USER program.
INPUT (RETURN)	Queued	No Action	Saved (immediate return)	Conversational	Signals successful receipt of a block of input. The message is queued, an input buffer is allocated, and control is returned to next statement of REQUEST.
INPUT (RETURN, NO BUFFER)	Queued	No Action	Saved (immediate return)	Receive	Signals successful receipt of input message but not completion of line discipline. Message is queued. No new buffer allocated. No action taken on attached output messages. Control returns to next statement of REQUEST. Intended for overlapping of processing of input message with acknowledgement of receipt by the REQUEST.
LOGICALACK	Queued	Lost	Saved	Receive	REQUEST is re-entered with the statement following the TERMINATE LOGICALACK upon command of the MCS.
NOINPUT	Lost	Re-queued	Lost	Receive	Signals unsuccessful attempt to receive input without errors. Input buffer returned and any output buffer is re-queued. The REQUEST ends.
OUTPUT	Lost	Lost	Lost	Transmit	Upon successful completion of output line discipline, output buffer and/or attached input buffers are returned. The REQUEST ends.

Table 3-3. Terminate Options (Cont)

TERMINATE OPTION	INPUT MESSAGE	OUTPUT MESSAGE	STATE	REQUEST TYPE	REMARKS
OUTPUT (RETURN)	Lost	Lost	Saved (CONTROL must continue)	Conversational	Signals successful transmission of a block of output data but not completion of the line discipline. Old output buffer is returned with no action taken on any attached input buffer. The state of the REQUEST is saved and is reinstated in the LINE CONTROL by a CONTINUE, defined for LINE(CONTROL KEY) = 4.
RELEASE STATION	Lost	Queued	Lost	Receive or Transmit	Makes the station available to all lines. Does not disconnect the line. Sets STATION (LINE) = 0.

LINE CONTROL SECTION

Line Control Section

Syntax:

```
< LINE CONTROL SECTION > ::= < CONTROL LIST >

< CONTROL LIST > ::= < CONTROL > | < CONTROL > , < CONTROL LIST >

< CONTROL > ::= CONTROL < CONTROL IDENTIFIER > : < CONTROL STATEMENT LIST >

< CONTROL IDENTIFIER > ::= < IDENTIFIER >

< CONTROL STATEMENT LIST > ::= < CONTROL STATEMENT >
                               | < CONTROL STATEMENT > . < CONTROL STATEMENT
                               LIST >

< CONTROL STATEMENT > ::= < ASSIGNMENT STATEMENT >
                          | < CASE STATEMENT >
                          | < CONTINUE STATEMENT >
                          | < DISPLAY STATEMENT >
                          | < DO STATEMENT >
                          | < DUMP STATEMENT >
                          | < IF STATEMENT >
                          | < INITIATE STATEMENT >
                          | < NULL STATEMENT >
                          | < POLL STATEMENT >
                          | < UNDO STATEMENT >
```

Semantics:

The user-written CONTROL is invoked by the Network Controller to decide which I/O function should get the use of a line. Each line in an NDL Data Comm System must have a line control procedure associated with it. The mandatory < LINE CONTROL STATEMENT > in the LINE Section handles this requirement.

The functions of a CONTROL ARE:

- a. Creation of a poll list.
- b. Choose a station on the line by testing System Status Variables such as STATION (QUEUED), FREQUENCY (INPUT), LINE (CONTROL KEY), etc., or by polling.
- c. Initiation of or restarting of the REQUEST for the chosen station.

REQUESTS are treated as I/O subroutines by CONTROLS, and it is the CONTROL that decides when and for which station a given REQUEST is to be executed. The REQUEST is invoked by the < CONTROL INITIATE STATEMENT >.

CONTROLS may never save state and suspend themselves as REQUESTS sometimes do. They are always entered at the first instruction of the CONTROL and run until they execute an < INITIATE STATEMENT > or a < CONTINUE STATEMENT >, at which time the CONTROL is terminated in favor of the Network Controller or the appropriate REQUEST.

The initial function of a CONTROL is to determine the state of the line by checking the value of LINE (CONTROL KEY). Certain values of LINE (CONTROL KEY) limit the range of action of the CONTROL or force the CONTROL to take a specific course of action. To simplify the coding of complex line controls, it is recommended that they be written as large CASE statements, using LINE (CONTROL KEY) as the CASE < EXPRESSION >.

Six of the CONTROL statements are virtually identical to REQUEST statements of the same name; they are the ASSIGNMENT, CASE, DO, IF, NULL, and UNDO statements. The CONTROL INITIATE statement serves a different purpose than the INITIATE statement in the REQUEST Section.

The NDL Compiler maintains a set of standard CONTROLS to handle common line driving techniques. Refer to Section 5, USING THE NDL COMPILER, in this manual for detailed information concerning these routines.

ASSIGNMENT Statement

Syntax:

⟨ ASSIGNMENT STATEMENT ⟩ ::= ⟨ ASSIGNABLE VARIABLE ⟩ := ⟨ EXPRESSION ⟩

⟨ ASSIGNABLE VARIABLE ⟩ ::= (See Basic Components).

Semantics:

The ASSIGNMENT statement is used to change the value of an assignable variable. The variable must be referenceable and assignable in CONTROLS.

Example:

```
STATION := STATION + 1.
```

CASE Statement

Syntax:

⟨ CASE STATEMENT ⟩ ::= ⟨ CASE HEAD ⟩
⟨ CASE BODY ⟩

⟨ CASE HEAD ⟩ ::= CASE ⟨ EXPRESSION ⟩

⟨ CASE BODY ⟩ ::= ⟨ CONTROL STATEMENT LIST ⟩
⟨ CASE ENDING ⟩

⟨ CASE ENDING ⟩ ::= END CASE

Semantics:

The ⟨ EXPRESSION ⟩ serves as an index into the list of ⟨ CONTROL STATEMENT ⟩ s. Only the selected statement is executed, and control then goes to the statement following the ⟨ CASE ENDING ⟩, unless the statement is an UNDO.

If there are n number of statements in the list, the range of ⟨ EXPRESSION ⟩ may be from 0 through n-1.

The statements in the list may be any legal ⟨ CONTROL STATEMENT ⟩ allowed in NDL. If the user wants to execute “nothing” in a given case, the NULL STATEMENT is appropriate.

CONTINUE Statement

Syntax:

⟨ CONTINUE STATEMENT ⟩ ::= CONTINUE

LINE CONTROL SECTION

Semantics:

The \langle CONTINUE STATEMENT \rangle is used in CONTROLS in conjunction with the TERMINATE OUTPUT (RETURN) statement in REQUESTS. Upon execution it returns control to the instruction following the previous TERMINATE OUTPUT (RETURN) statement in the REQUEST.

DISPLAY Statement

Syntax:

\langle DISPLAY STATEMENT $\rangle ::=$ DISPLAY \langle EXPRESSION LIST \rangle

\langle EXPRESSION LIST $\rangle ::=$ \langle EXPRESSION \rangle | \langle EXPRESSION \rangle , \langle EXPRESSION LIST \rangle

Semantics:

The DISPLAY statement allows a message to be printed on the console printer. \langle EXPRESSION \rangle must be a literal, constant identifier, or variable name.

Example:

```
DISPLAY "LINE", DECIMAL(LINE,1), " , STATION",  
DECIMAL (STATION,1), " IS NOT RESPONDING".
```

When the DISPLAY statement shown in the example above is executed, the following output results on the console printer:

```
LINE 1 , STATION 3 IS NOT RESPONDING
```

DO Statement

Syntax:

\langle DO STATEMENT $\rangle ::=$ \langle DO HEAD \rangle \langle CONTROL STATEMENT LIST \rangle
 \langle END PART \rangle

\langle DO HEAD $\rangle ::=$ DO \langle IDENTIFIER PART \rangle \langle FOREVER PART \rangle .

\langle END PART $\rangle ::=$ END \langle IDENTIFIER PART \rangle .

\langle IDENTIFIER PART $\rangle ::=$ \langle EMPTY \rangle | \langle IDENTIFIER \rangle

\langle FOREVER PART $\rangle ::=$ \langle EMPTY \rangle | FOREVER

Semantics:

The `< DO STATEMENT >` provides a means of grouping two or more CONTROL statements (including other `< DO STATEMENT >` s) as one statement. This is useful in `< IF STATEMENTS >` and `< CASE STATEMENTS >` . `< DO STATEMENT >` s may be nested. The `< DO IDENTIFIER >` is optional.

Example:

```

IF TOG [1] THEN
    DO OUTER.
        DO INNER FOREVER.
            INITIATE INPUT.
            UNDO INNER.           % This UNDO would never be executed because
                                % INITIATE INPUT causes program control
                                % to go to the appropriate REQUEST.
        END INNER.
        TALLY [1] := TALLY [1] + 1.
    END OUTER.
```

When the `< FOREVER PART >` is used it causes the `< CONTROL STATEMENT LIST >` to be executed repeatedly until an UNDO is executed, or a statement such as INITIATE INPUT is executed that causes control to exit the `< DO STATEMENT >` .

Restrictions:

1. If a `< DO IDENTIFIER >` is included in the `< DO HEAD >` , it must also be included in the `< END PART >` .
2. If the `< DO HEAD >` does not include a `< DO IDENTIFIER >` , the `< END PART >` must not include one.

DUMP Statement

Syntax:

```
< DUMP STATEMENT > ::= DUMP
```

Semantics:

When DUMP is specified, the contents of a program's memory space is dumped to disk for subsequent analysis by NDL/DUMP. Processing automatically continues when the dump is finished.

IF Statement

Syntax:

```
< IF STATEMENT > ::= IF < EXPRESSION > THEN < CONTROL STATEMENT > .
                       < ELSE PART >
```

```
< ELSE PART > ::= ELSE < CONTROL STATEMENT > | < EMPTY >
```

Semantics:

The `< EXPRESSION >` is evaluated, and if the least significant bit of the value is equal to one (TRUE) the statement following THEN will be executed. If the bit is off (FALSE) the statement following ELSE (if present) will be executed. Control always passes to the statement following the IF statement. Any `< CONTROL STATEMENT >` may follow THEN or ELSE, including `< DO STATEMENT >` s, `< IF STATEMENT >` s, and `< NULL STATEMENT >` s.

LINE CONTROL SECTION

INITIATE Statement

Syntax:

```
< INITIATE STATEMENT > ::= INITIATE < INITIATE OPTION >  
< INITIATE OPTION > ::= < I-O OPTION > | < AUTOPOLL OPTION >  
                        | < IDLE OPTION > | < CANCEL OPTION > | < DISCONNECT OPTION >  
< I-O OPTION > ::= INPUT | OUTPUT | INPUTOUTPUT | OUTPUTINPUT  
                | INPUT (NO BUFFER) | OUTPUT (NO BUFFER)  
< AUTOPOLL OPTION > ::= AUTOPOLL < AUTOPOLL SLAVE ADDRESS >  
< AUTOPOLL SLAVE ADDRESS > ::= (< INTEGER >) | < EMPTY >  
< IDLE OPTION > ::= IDLE  
< CANCEL OPTION > ::= CANCEL  
< DISCONNECT OPTION > ::= DISCONNECT
```

Semantics:

The < INITIATE STATEMENT > ends the CONTROL, and depending on the < INITIATE OPTION >, either starts an I/O REQUEST, cancels an I/O in progress, starts hardware autopoll, or idles the line. The < INITIATE OPTION > executed must be appropriate to the current line and current station status. For instance, INITIATE OUTPUT will cause a fatal run error if there is no message queued for the current station.

It is the NDL programmer's responsibility to either test all relevant conditions before executing an INITIATE or to structure his NDL program so that the INITIATE cannot be inappropriate.

The descriptions of the < INITIATE OPTION > s follow:

I-O OPTION. All I-O OPTION variants begin a REQUEST, the REQUEST chosen depending upon the variant and the station's < TERMINAL REQUEST STATEMENT > as defined in the TERMINAL Section.

INITIATE INPUT starts the input REQUEST, attaches an input buffer, and sets the INPUTATTACHED boolean TRUE. The station must be valid, ready, enabled, and STATION (MYUSE) must permit input.

Example:

```
IF STATION (VALID) AND STATION (READY) AND  
   STATION (ENABLED) AND STATION (MYUSE) NE 2  
THEN INITIATE INPUT.
```

INITIATE INPUT (NO BUFFER) starts the input REQUEST, but does not attach an input buffer. The station must be valid, ready, enabled, and STATION (MYUSE) must permit input. If an input buffer is desired later, the REQUEST must execute a TERMINATE INPUT (RETURN).

INITIATE OUTPUT starts the output REQUEST, attaches an output message, and sets the OUTPUT-ATTACHED boolean TRUE. The current station must be valid, ready, queued, and STATION (MYUSE) must allow output.

INITIATE OUTPUT (NO BUFFER) starts the output REQUEST, but no output message is attached even though one must be queued for output in order to execute this INITIATE. To get the message while running, the REQUEST must execute a TERMINATE OUTPUT (RETURN).

INITIATE INPUTOUTPUT starts the input REQUEST, while INITIATE OUTPUTINPUT starts the output REQUEST. Otherwise, they have the same function, both getting an input buffer and an output message and both setting INPUTATTACHED and OUTPUTATTACHED TRUE. The station must be enabled, valid, ready, and queued, and STATION (MYUSE) must equal three (input/output). These initiates are used to start conversational line disciplines.

AUTOPOLL OPTION takes the poll string which is built by the POLL statement, and initiates an I/O which starts recirculating polling (AUTOPOLL). Polling continues automatically until a message is received or an error occurs. Then STATION is set to the proper value and the input REQUEST is entered, as if an INITIATE INPUT had been executed. The REQUEST should check for I/O Exception. Re-initiating AUTOPOLL causes polling to resume where it stopped previously. The poll string must be rebuilt to add or delete stations.

⟨ AUTOPOLL SLAVE ADDRESS ⟩ is required only when the station may reply with an address that is different than the polled address, as in controller/slave station configurations. The ⟨ INTEGER ⟩ specifies the position of the first character of the slave station address in the message received.

NOTE

The numbering of the characters in the message begins with one (1), so that if the address begins at the third character of the message the ⟨ INTEGER ⟩ should be a three (“3”).

The IDLE OPTION is used to place the line in a state of semi-permanent rest (idled). The line is marked “not busy” but remains in ready state. While the line is idle the LINE CONTROL will not be entered, so the line will not respond to stations that attempt input. Any messages that were queued for output on the line at the time it was idled will not be transmitted. This command may be used to ease the load on the controller by deactivating a line on which no traffic is expected. The line can be reactivated by one of these three events:

- a. Execution by the MCS of a CHANGE command to the Network Controller.
- b. Opening of a file by an application program involving a station on this line, if no MCS is present.
- c. Queuing of an output message for a station on this line when the station has no other messages queued.

The CANCEL OPTION is used to halt any receive operation in progress on a line if no data has been received. If the receive operation was either an INITIATE RECEIVE (NO TIMEOUT) or a FINISH TRANSMIT-INITIATE RECEIVE (NO TIMEOUT), the CANCEL is effective if either: (a) less than 12 characters have been received and the I/O control is a Single Line Control, or (b) no characters have been received and the I/O control is a Multi-Line Control.

Logically, only one task at a time may be performed on a line. The line control procedure is not normally entered while an I/O operation is in progress on the line, but the following exceptions may occur:

- a. If an INITIATE AUTOPOLL, INITIATE RECEIVE (NO TIMEOUT), or FINISH TRANSMIT-INITIATE RECEIVE (NO TIMEOUT) are in progress and no messages are queued for the line, the line control procedure is entered once every 30 seconds to allow the user the option of canceling the receive operation currently in progress.

LINE CONTROL SECTION

- b. A message is queued for output in contention with one of the three receive operations (listed in paragraph (a.) in progress.
- c. The receive operation may be stopped with an INITIATE CANCEL when the line control procedure is entered and LINE(CONTROL KEY) is equal to 2. After canceling the receive operation, the Network Controller forces LINE(CONTROL KEY) equal to 0 before re-entering the CONTROL.

The DISCONNECT OPTION of the INITIATE statement may be executed in CONTROL procedures to disconnect (“hang up”) a line. The appropriate time to use this option is after determining that all of the stations on the line no longer require service (not ENABLED, not performing output, or timing out).

NULL Statement

Syntax:

⟨ NULL STATEMENT ⟩ ::= ⟨ EMPTY ⟩

Semantics:

The NULL statement can be used in an IF statement or a CASE statement where an executable statement is required but a “no-operation” is desired.

Example:

```
CASE TALLY [0].
  IF TOG [0] THEN           %           CASE entry 0.
                           %  NULL.
  ELSE
    INITIATE INPUT.        %           CASE entry 1.
  DO.                       %           CASE entry 2.
    ⟨ REQUEST STATEMENT LIST ⟩
  END.
END CASE.
```

In the example, the CASE contains three executable statements, an IF, a NULL, and a DO. One of the three is executed depending on the value of TALLY [0]. The possibilities are:

- a. If TALLY [0] is equal to zero, the IF statement will be executed. If the THEN branch is taken, a NULL results and the CASE statement is exited.
- b. If TALLY [0] is equal to one the NULL statement is executed and the CASE is exited.
- c. If TALLY [0] is equal to two, the DO statement is executed.

POLL Statement

Syntax:

⟨ POLL STATEMENT ⟩ ::= POLL ⟨ POLL ELEMENT LIST ⟩

⟨ POLL ELEMENT LIST ⟩ ::= ⟨ POLL ELEMENT ⟩
| ⟨ POLL ELEMENT ⟩, ⟨ POLL ELEMENT LIST ⟩

⟨ POLL ELEMENT ⟩ ::= ⟨ CONSTANT IDENTIFIER ⟩
| ⟨ STRING ⟩
| ⟨ ADDRESS ⟩

⟨ ADDRESS ⟩ ::= ADDRESS
| ADDRESS (TRANSMIT)

Semantics:

The < POLL STATEMENT > is used to create a poll string before initiating AUTOPOLL (Refer to the < AUTOPOLL OPTION > of the CONTROL INITIATE statement). The CONTROL should cycle through the stations on the line, executing a POLL statement once for each station remaining in the poll string. A < POLL ELEMENT LIST > must contain exactly one occurrence of < ADDRESS >. ADDRESS and ADDRESS (TRANSMIT) are equivalent in the POLL statement.

Restrictions:

- a. Each < POLL ELEMENT LIST > entered in the same poll string must have the same total length.
- b. The maximum length of a < POLL ELEMENT LIST > is 15 characters.
- c. The total length of the poll string should not exceed the length specified in the AUTOPOLL SIZE statement in the LINE Section. ENDOFBUFFER will be set TRUE if the length is exceeded.
- d. The < POLL STATEMENT > should be executed only for stations that are ready, valid, and enabled. Also, the MYUSE statement must specify INPUT.
- e. Group Poll: If the Data Comm system includes Controller/Slave Stations, STATION (TYPE) should be checked to avoid polling Slave Stations.

Example:

```

STATION := 1.
DO FOREVER.
  IF STATION (VALID) AND STATION (READY) AND
    STATION (ENABLED) AND (STATION(MYUSE) NE 2) AND
    (STATION(TYPE) NE 1)
    THEN POLL EOT, ADDRESS, POL, 4 "2E".
  IF ENDOFBUFFER THEN UNDO.
  IF STATION EQ MAXSTATIONS THEN UNDO.
  STATION := STATION + 1.
END.

```

LINE CONTROL SECTION

UNDO Statement

Syntax:

⟨ UNDO STATEMENT ⟩ ::= UNDO ⟨ DO OPTION ⟩

⟨ DO OPTION ⟩ ::= ⟨ EMPTY ⟩ | ⟨ DO IDENTIFIER ⟩

Semantics:

The ⟨ UNDO STATEMENT ⟩ is used to exit a ⟨ DO STATEMENT ⟩. Execution of the UNDO will cause control to transfer to the statement following the ⟨ END PART ⟩ of the ⟨ DO STATEMENT ⟩.

The ⟨ DO OPTION ⟩ may take one of two forms:

- a. UNDO transfers control out of the innermost DO statement in which it appears.
- b. UNDO ⟨ DO IDENTIFIER ⟩ transfers control out of the ⟨ DO STATEMENT ⟩ with that ⟨ DO IDENTIFIER ⟩.

Example:

```
DO OUTER.  
  DO.  
    IF TOG [0] THEN UNDO.           %   Exits the DO ⟨EMPTY⟩  
    DO INNER.  
      IF TOG [1] THEN UNDO.       %   Exits "INNER"  
      UNDO OUTER.                 %   Exits "OUTER"  
    END INNER.  
  END.  
END OUTER.
```

Terminal Section

Syntax:

⟨ TERMINAL SECTION ⟩ :: = ⟨ TERMINAL LIST ⟩

⟨ TERMINAL LIST ⟩ :: = ⟨ TERMINAL DEFINITION ⟩
 | ⟨ TERMINAL DEFINITION ⟩ ⟨ TERMINAL LIST ⟩
 | ⟨ TERMINAL DEFAULT DEFINITION ⟩ ⟨ TERMINAL LIST ⟩

⟨ TERMINAL DEFINITION ⟩ :: = TERMINAL ⟨ TERMINAL IDENTIFIER ⟩ :
 ⟨ TERMINAL ATTRIBUTE LIST ⟩

⟨ TERMINAL DEFAULT DEFINITION ⟩ :: = TERMINAL DEFAULT ⟨ TERMINAL DEFAULT
 IDENTIFIER ⟩ :
 ⟨ TERMINAL ATTRIBUTE LIST ⟩ .

⟨ TERMINAL ATTRIBUTE LIST ⟩ :: = ⟨ TERMINAL ATTRIBUTE STATEMENT ⟩ .
 | ⟨ TERMINAL ATTRIBUTE STATEMENT ⟩ .
 ⟨ TERMINAL ATTRIBUTE LIST ⟩

⟨ TERMINAL ATTRIBUTE STATEMENT ⟩ :: = ⟨ BUFFERSIZE STATEMENT
 | ⟨ TERMINAL ADDRESS STATEMENT ⟩
 | ⟨ TERMINAL DEFAULT STATEMENT ⟩
 | ⟨ TERMINAL DIAGNOSTIC REQUEST
 STATEMENT ⟩
 | ⟨ TERMINAL REQUEST STATEMENT ⟩
 | ⟨ TERMINAL TYPE STATEMENT ⟩
 | ⟨ TRANSMISSION NUMBER STATEMENT ⟩

⟨ TERMINAL IDENTIFIER ⟩ :: = ⟨ IDENTIFIER ⟩

⟨ TERMINAL DEFAULT IDENTIFIER ⟩ :: = ⟨ IDENTIFIER ⟩

Restrictions:

1. At least one ⟨ TERMINAL DEFINITION ⟩ must appear in the TERMINAL Section of every NDL program.
2. A ⟨ TERMINAL DEFAULT DEFINITION ⟩ must precede any reference to it in a ⟨ TERMINAL DEFAULT STATEMENT ⟩ .
3. The ⟨ TERMINAL DEFAULT STATEMENT ⟩ may appear in a ⟨ TERMINAL DEFINITION ⟩ but not in a ⟨ TERMINAL DEFAULT DEFINITION ⟩ .

Semantics:

The purpose of the TERMINAL Section is to describe each different type of remote device that is connected to the Data Comm system. If the system consists of TC 500 and TD 700 devices, for example, two terminal definitions are required in the TERMINAL Section, one for each type of device.

Three of the ⟨ TERMINAL ATTRIBUTE STATEMENT ⟩ s are used to link this section to the other sections of the NDL program. Terminals are associated with REQUESTS by the ⟨ TERMINAL REQUEST STATEMENT ⟩ and the ⟨ TERMINAL DIAGNOSTIC REQUEST STATEMENT ⟩ . A station is associated with a terminal definition by the ⟨ TERMINAL STATEMENT ⟩ in the STATION Section.

TERMINAL SECTION

Of the TERMINAL ATTRIBUTE STATEMENTS, only the \langle TERMINAL REQUEST STATEMENT \rangle and the \langle BUFFERSIZE STATEMENT \rangle are required in each terminal definition. However, the NDL Compiler will print a warning on the listing of the object program if the TERMINAL ADDRESS, HEADER, TRANSMISSION NUMBER, or TERMINAL TYPE statements are omitted.

For terminals which have several common attributes, the common attributes may be grouped in a \langle TERMINAL DEFAULT DEFINITION \rangle . The remaining attributes can be listed explicitly under each individual \langle TERMINAL DEFINITION \rangle that also includes the TERMINAL DEFAULT STATEMENT.

BUFFERSIZE Statement

Syntax:

\langle BUFFERSIZE STATEMENT $\rangle ::=$ BUFFERSIZE = \langle INTEGER \rangle

Semantics:

This statement defines the maximum size of a buffer in bytes and applies to inputs from and outputs to the terminal. This statement is required in all terminal definitions. The \langle INTEGER \rangle must be less than 4096.

TERMINAL ADDRESS Statement

Syntax:

\langle TERMINAL ADDRESS STATEMENT $\rangle ::=$ ADDRESS = \langle RECEIVE ADDRESS \rangle
 \langle TRANSMIT ADDRESS PART \rangle
| ADDRESS = NULL

\langle RECEIVE ADDRESS $\rangle ::=$ \langle ADDRESS SIZE \rangle

\langle TRANSMIT ADDRESS PART $\rangle ::=$ \langle TRANSMIT ADDRESS \rangle | \langle EMPTY \rangle

\langle TRANSMIT ADDRESS $\rangle ::=$ \langle ADDRESS SIZE \rangle

\langle ADDRESS SIZE $\rangle ::=$ 1 | 2 | 3

Semantics:

This statement specifies the number of characters that are used to address a station associated with a terminal. If the \langle TRANSMIT ADDRESS PART \rangle is \langle EMPTY \rangle it is implied that the \langle TRANSMIT ADDRESS \rangle is the same length as the \langle RECEIVE ADDRESS \rangle . If the \langle TRANSMIT ADDRESS PART \rangle is included it indicates that the \langle TRANSMIT ADDRESS \rangle may be non-identical to the \langle RECEIVE ADDRESS \rangle .

If the \langle TERMINAL ADDRESS STATEMENT \rangle is omitted, both address lengths are assumed to be zero.

TERMINAL DEFAULT Statement

Syntax:

\langle TERMINAL DEFAULT STATEMENT $\rangle ::=$ DEFAULT = \langle TERMINAL DEFAULT IDENTIFIER \rangle

Semantics:

This statement specifies the \langle TERMINAL DEFAULT IDENTIFIER \rangle of a preceding \langle TERMINAL DEFAULT DEFINITION \rangle . The attributes in the \langle TERMINAL DEFAULT DEFINITION \rangle are added to the explicit attributes specified in the \langle TERMINAL DEFINITION \rangle in which this statement appears.

Restrictions:

1. If a default attribute conflicts with an explicit attribute in the applicable \langle TERMINAL DEFINITION \rangle , the explicit attribute is given precedence.
2. The \langle TERMINAL DEFAULT STATEMENT \rangle may not appear in a \langle TERMINAL DEFAULT DEFINITION \rangle .
3. A \langle TERMINAL DEFAULT DEFINITION \rangle must precede any reference to it in a \langle TERMINAL DEFAULT STATEMENT \rangle .

Example:

Assume that terminals A1, A2, and A3 have the same attributes. As illustrated below, a \langle TERMINAL DEFAULT STATEMENT \rangle could be used to simplify the coding of the descriptions of these three terminals.

```

TERMINAL DEFAULT A123DFLT:
    REQUEST    =    READTTY : RECEIVE, WRITETTY : TRANSMIT.
    MAXINPUT   =    72.
    ADDRESS    =    NULL.
TERMINAL A1:
    DEFAULT    =    A123DFLT.
    MAXINPUT   =    95.
TERMINAL A2:
    DEFAULT    =    A123DFLT.
    ADDRESS    =    2.
TERMINAL A3:
    DEFAULT    =    A123DFLT.

```

TERMINAL DIAGNOSTIC REQUEST Statement

Syntax:

```

 $\langle$ TERMINAL DIAGNOSTIC REQUEST STATEMENT $\rangle$  ::= DIAGNOSTIC =
                                            $\langle$ REQUEST SPECIFIER LIST $\rangle$ 
 $\langle$ REQUEST SPECIFIER LIST $\rangle$  ::=  $\langle$ INPUT SPECIFIER $\rangle$ 
                               | $\langle$ OUTPUT SPECIFIER $\rangle$ 
                               | $\langle$ INPUT SPECIFIER $\rangle$  ,  $\langle$ OUTPUT SPECIFIER $\rangle$ 
                               | $\langle$ OUTPUT SPECIFIER $\rangle$  ,  $\langle$ INPUT SPECIFIER $\rangle$ 
 $\langle$ INPUT SPECIFIER $\rangle$  ::=  $\langle$ REQUEST IDENTIFIER $\rangle$  : RECEIVE
 $\langle$ OUTPUT SPECIFIER $\rangle$  ::=  $\langle$ REQUEST IDENTIFIER $\rangle$  : TRANSMIT

```

Semantics:

This optional statement specifies an alternate REQUEST that can be used for diagnostic purposes. A CHANGE from the MCS is required to initiate the diagnostic routines.

TERMINAL SECTION

TERMINAL REQUEST Statement

Syntax:

⟨ TERMINAL REQUEST STATEMENT ⟩ ::= REQUEST = ⟨ REQUEST SPECIFIER LIST ⟩

⟨ REQUEST SPECIFIER LIST ⟩ ::= ⟨ INPUT SPECIFIER ⟩
 | ⟨ OUTPUT SPECIFIER ⟩
 | ⟨ INPUT SPECIFIER ⟩, ⟨ OUTPUT SPECIFIER ⟩
 | ⟨ OUTPUT SPECIFIER ⟩, ⟨ INPUT SPECIFIER ⟩

⟨ INPUT SPECIFIER ⟩ ::= ⟨ REQUEST IDENTIFIER ⟩ : RECEIVE

⟨ OUTPUT SPECIFIER ⟩ ::= ⟨ REQUEST IDENTIFIER ⟩ : TRANSMIT

Semantics:

A ⟨ TERMINAL REQUEST STATEMENT ⟩ is required in every ⟨ TERMINAL DEFINITION ⟩, and links the terminal with one or two line disciplines.

The ⟨ REQUEST SPECIFIER LIST ⟩ may specify an input REQUEST, an output REQUEST, or both. The same REQUEST can be used for both input and output.

Example:

REQUEST = CANDEIOTTY:RECEIVE, CANDEIOTTY:TRANSMIT.

When input is to be received from a station, that station's terminal-table entry is checked and the REQUEST identified in the ⟨ INPUT SPECIFIER ⟩ is called. For output, the REQUEST identified in the ⟨ OUTPUT SPECIFIER ⟩ is called.

TERMINAL TYPE Statement

Syntax:

< TERMINAL TYPE STATEMENT > ::= TYPE = < TYPE NUMBER >

< TYPE NUMBER > ::= < INTEGER >

Semantics:

On input messages, an eight-bit field in the message header is automatically set by the Network Controller to the terminal TYPE specified for the current station.

The value of < TYPE NUMBER > must be less than 64. The values currently assigned for < TYPE NUMBER > are defined below, and are passed in the message header.

<u>Value</u>	<u>Terminal TYPE</u>
0	TTY
1	TC500
2	B9352 (Wide)
3	RT2000
5	TU300
6	TC3500
7	DC140
8	TC4000
9	TC5100
10	TT102
11	TC700
12	B9352 (Narrow)
13	DC110
14	B9350
15	TU400
21	TC1700
22	B9353
25	TU500
32	TD700
35	TU700
42	TD800
44	TD820
45	TU800
62	B1800/B1700
63	B6700

If a < TERMINAL TYPE STATEMENT > does not appear in a < TERMINAL DEFINITION > a compiler warning will be printed and < TYPE NUMBER > will be set to zero.

TRANSMISSION NUMBER Statement

Syntax:

< TRANSMISSION NUMBER STATEMENT > ::= TRANSMISSION = < TRANSMISSION NUMBER >

< TRANSMISSION NUMBER > ::= 0 | 1 | 2 | 3 | NULL

Semantics:

The length of the transmission number is set by this statement. The number 0 and NULL are equivalent and indicate that no transmission number will be used. If the < ITEM > TRAN (refer to the INITIALIZE STATEMENT in the REQUEST Section) is mentioned in a REQUEST of the terminal, a non-null transmission number must be used.

STATION SECTION

Station Section

Syntax:

⟨ STATION SECTION ⟩ ::= ⟨ STATION LIST ⟩

⟨ STATION LIST ⟩ ::=
⟨ STATION DEFINITION ⟩
| ⟨ STATION DEFINITION ⟩ ⟨ STATION LIST ⟩
| ⟨ STATION DEFAULT DEFINITION ⟩ ⟨ STATION LIST ⟩

⟨ STATION DEFINITION ⟩ ::= STATION ⟨ STATION IDENTIFIER ⟩ : ⟨ STATION ATTRIBUTE LIST ⟩

⟨ STATION DEFAULT DEFINITION ⟩ ::= STATION DEFAULT ⟨ STATION DEFAULT IDENTIFIER ⟩ : ⟨ STATION ATTRIBUTE LIST ⟩

⟨ STATION ATTRIBUTE LIST ⟩ ::=
⟨ STATION ATTRIBUTE STATEMENT ⟩ .
| ⟨ STATION ATTRIBUTE STATEMENT ⟩ .
| ⟨ STATION ATTRIBUTE LIST ⟩

⟨ STATION ATTRIBUTE STATEMENT ⟩ ::=
⟨ CONTROLLER STATEMENT ⟩
| ⟨ FREQUENCY STATEMENT ⟩
| ⟨ LOGICALACK STATEMENT ⟩
| ⟨ MYUSE STATEMENT ⟩
| ⟨ RETRY STATEMENT ⟩
| ⟨ STATION ADDRESS STATEMENT ⟩
| ⟨ STATION DEFAULT STATEMENT ⟩
| ⟨ STATION READY STATEMENT ⟩
| ⟨ STATION TERMINAL STATEMENT ⟩

⟨ STATION IDENTIFIER ⟩ ::= ⟨ IDENTIFIER ⟩

⟨ STATION DEFAULT IDENTIFIER ⟩ ::= ⟨ IDENTIFIER ⟩

Semantics:

The STATION Section is used to define each individual remote device in an NDL Data Comm system. Default definitions are provided to avoid repetitious coding of attributes, as in the TERMINAL Section.

Restrictions:

1. The STATION Section must contain at least one ⟨ STATION DEFINITION ⟩ .
2. A ⟨ STATEMENT DEFAULT DEFINITION ⟩ must precede any reference to it in a ⟨ STATION DEFAULT STATEMENT ⟩ .
3. The ⟨ STATION DEFAULT STATEMENT ⟩ may appear in a ⟨ STATION DEFINITION ⟩ but not in a ⟨ STATION DEFAULT DEFINITION ⟩ . (Defaults may not be nested.)

CONTROLLER Statement

Syntax:

⟨ CONTROLLER STATEMENT ⟩ ::= CONTROLLER = ⟨ STATION CONTROLLER TYPE ⟩

```

< STATION CONTROLLER TYPE > ::= TRUE
                               | FALSE
                               | < STATION IDENTIFIER >

```

Semantics:

This statement is required only for stations that are a part of a Controller/Slave Station configuration, as illustrated in Figure 3-1. The < CONTROLLER STATEMENT > makes it possible to use group polling in the LINE CONTROL (refer to the < POLL STATEMENT > in the LINE CONTROL Section). A station's CONTROLLER status can be accessed at run-time by using the variable STATION (TYPE).

In Figure 3-1, the TU910 would be designated a Controller for the four TU510 Slave Stations.

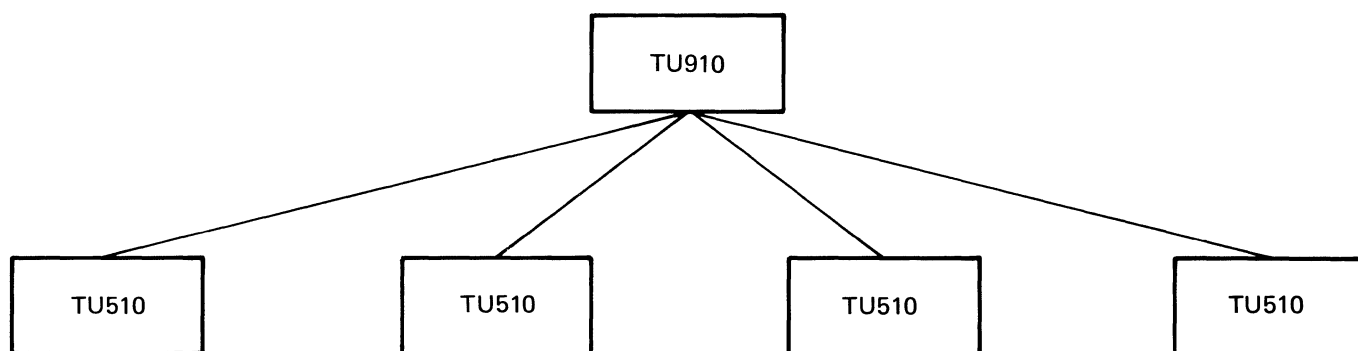


Figure 3-1. Controller/Slave Stations

The three < STATION CONTROLLER TYPE > s are described as follows:

TRUE - This station is a Controller, such as the TU910 in Figure 3-1.

FALSE - This is the default value, and indicates that the station is a regular remote device.

< STATION IDENTIFIER > - Indicates that this is a Slave Station, such as the TU510's in Figure 3-1, and < STATION IDENTIFIER > is its Controller. < STATION IDENTIFIER > must refer to a preceding < STATION DEFINITION > .

FREQUENCY Statement

Syntax:

```

< FREQUENCY STATEMENT > ::= FREQUENCY = < INTEGER > , < INTEGER >
                               | < INTEGER >

```

Semantics:

This optional statement specifies a priority for input and output for use by the station in the LINE CONTROL section. < INTEGER > may be a maximum of 255.

STATION SECTION

The first integer defines the input priority and the second integer defines the output priority. The default value is zero.

Examples:

```
FREQUENCY = 3, 7.   % INPUT = 3, OUTPUT = 7
FREQUENCY = 3.     % INPUT = 3, OUTPUT = 0
```

LOGICALACK Statement

Syntax:

⟨ LOGICALACK STATEMENT ⟩ ::= LOGICALACK = ⟨ LOGICAL VALUE ⟩

Semantics:

If the LOGICAL VALUE is "TRUE" in the NDL program, the REQUEST will actually be interrupted. LOGICALACK defaults to "FALSE" if this statement is omitted from the STATION DEFINITION.

Refer to the TERMINATE LOGICALACK statement in the REQUEST Section for additional information about LOGICALACK.

MYUSE Statement

Syntax:

⟨ MYUSE STATEMENT ⟩ ::= MYUSE = ⟨ IO LIST ⟩

⟨ IO LIST ⟩ ::= INPUT
 | OUTPUT
 | INPUT, OUTPUT
 | OUTPUT, INPUT

Semantics:

This statement indicates the functional use of a station; INPUT, OUTPUT, or both. It should be consistent with the ⟨ TERMINAL REQUEST STATEMENT ⟩ of this station's ⟨ TERMINAL DEFINITION ⟩ (refer to the TERMINAL Section of this document), and with the I/O initiation in the LINE CONTROL (refer to the ⟨ INITIATE STATEMENT ⟩ in the LINE CONTROL Section).

The possible values for STATION(MYUSE) are (according to function), as follows:

<u>Value</u>	<u>Function</u>
1	INPUT
2	OUTPUT
3	INPUT, OUTPUT, or OUTPUT, INPUT

The status of MYUSE can be checked at run time by accessing the variable STATION (MYUSE).

MYUSE must be defined for every ⟨ STATION DEFINITION ⟩.

The MCS can change a station's MYUSE value.

RETRY Statement

Syntax:

⟨ RETRY STATEMENT ⟩ ::= RETRY = ⟨ INTEGER ⟩

Semantics:

This statement provides the initial retry count for the station. The value given can be changed by a MCS.

The ⟨ INTEGER ⟩ must be less than 256.

The RETRY count can be reinitialized at run time by the INITIALIZE RETRY statement in a REQUEST.

STATION ADDRESS Statement

Syntax:

⟨ STATION ADDRESS STATEMENT ⟩ ::= ADDRESS = ⟨ STATION ADDRESS LIST ⟩

⟨ STATION ADDRESS LIST ⟩ ::= ⟨ RECEIVE-TRANSMIT ADDRESS ⟩
| ⟨ RECEIVE ADDRESS ⟩ , ⟨ TRANSMIT ADDRESS ⟩

⟨ RECEIVE-TRANSMIT ADDRESS ⟩ ::= ⟨ EBCDIC STRING ⟩

⟨ RECEIVE ADDRESS ⟩ ::= ⟨ EBCDIC STRING ⟩

⟨ TRANSMIT ADDRESS ⟩ ::= ⟨ EBCDIC STRING ⟩

Semantics:

The address of a station is specified by this statement for operations such as polling and selecting. If a ⟨ STATION ADDRESS STATEMENT ⟩ is not defined for a station, the access mode of the associated terminal must be contention.

The length of the address specified must be equal to the length as defined in the ⟨ TERMINAL ADDRESS STATEMENT ⟩ of the associated terminal in the TERMINAL Section.

If the stations ⟨ RECEIVE ADDRESS ⟩ and ⟨ TRANSMIT ADDRESS ⟩ are different, they must both be specified and separated by a comma. The ⟨ RECEIVE ADDRESS ⟩ must be specified first, the ⟨ TRANSMIT ADDRESS ⟩ second.

Example:

ADDRESS = "01".
ADDRESS = "S10", "S11".

STATION SECTION

STATION DEFAULT Statement

Syntax:

⟨ STATION DEFAULT STATEMENT ⟩ ::= DEFAULT = ⟨ STATION DEFAULT IDENTIFIER ⟩

Semantics:

This statement is used in the same manner and for a similar purpose as a ⟨ TERMINAL DEFAULT STATEMENT ⟩ is used in the TERMINAL Section.

Example:

```
STATION DEFAULT DF:                % STATION DEFAULT DEFINITION
MYUSE = INPUT, OUTPUT.
READY = FALSE
TERMINAL = TTY.
RETRY = 5.
FREQUENCY = 4, 8.
CONTROLLER = FALSE.
```

STATION S1:

```
DEFAULT = DF.
ADDRESS = "S1".
```

In the above example, all of the attributes specified in the ⟨ STATION DEFAULT DEFINITION ⟩ will also apply to STATION S1 because of the DEFAULT = DF statement in that station's definition.

STATION READY Statement

Syntax:

⟨ STATION READY STATEMENT ⟩ ::= READY = ⟨ LOGICAL VALUE ⟩

Semantics:

If ⟨ LOGICAL VALUE ⟩ is FALSE, it allows the Network Controller to complete the BOJ functions but does not allow it to poll this station. READY defaults to TRUE if this statement is omitted from the ⟨ STATION DEFINITION ⟩.

STATION TERMINAL Statement

Syntax:

⟨ STATION TERMINAL STATEMENT ⟩ ::= TERMINAL = ⟨ TERMINAL IDENTIFIER ⟩

Semantics:

This statement must be included in every ⟨ STATION DEFINITION ⟩, and it specifies the name of the ⟨ TERMINAL DEFINITION ⟩ which defines the physical characteristics of the station being described.

More than one ⟨ STATION DEFINITION ⟩ may reference the same ⟨ TERMINAL DEFINITION ⟩.

Line Section

Syntax:

⟨ LINE SECTION ⟩ ::= ⟨ LINE LIST ⟩

⟨ LINE LIST ⟩ ::= ⟨ LINE DEFINITION ⟩
 | ⟨ LINE DEFINITION ⟩ ⟨ LINE LIST ⟩
 | ⟨ LINE DEFAULT DEFINITION ⟩ ⟨ LINE LIST ⟩

⟨ LINE DEFINITION ⟩ ::= LINE ⟨ LINE IDENTIFIER ⟩ : ⟨ LINE ATTRIBUTE LIST ⟩

⟨ LINE DEFAULT DEFINITION ⟩ ::= LINE DEFAULT ⟨ LINE DEFAULT IDENTIFIER ⟩ :
 ⟨ LINE ATTRIBUTE LIST ⟩

⟨ LINE ATTRIBUTE LIST ⟩ ::= ⟨ LINE ATTRIBUTE STATEMENT ⟩ .
 | ⟨ LINE ATTRIBUTE STATEMENT ⟩ .
 ⟨ LINE ATTRIBUTE LIST ⟩

⟨ LINE ATTRIBUTE STATEMENT ⟩ ::= ⟨ AUTOPOLL SIZE STATEMENT ⟩
 | ⟨ LINE ADDRESS STATEMENT ⟩
 | ⟨ LINE CONTROL STATEMENT ⟩
 | ⟨ LINE DEFAULT STATEMENT ⟩
 | ⟨ LINE STATION STATEMENT ⟩
 | ⟨ LINE TYPE STATEMENT ⟩

⟨ LINE IDENTIFIER ⟩ ::= ⟨ IDENTIFIER ⟩

⟨ LINE DEFAULT IDENTIFIER ⟩ ::= ⟨ IDENTIFIER ⟩

Restrictions:

1. At least one ⟨ LINE DEFINITION ⟩ must appear in the LINE Section of every NDL program.
2. A ⟨ LINE DEFAULT DEFINITION ⟩ must precede a reference to it in a ⟨ LINE DEFAULT STATEMENT ⟩.
3. The ⟨ LINE DEFAULT STATEMENT ⟩ may not appear in a ⟨ LINE DEFAULT DEFINITION ⟩. Defaults may not be nested.
4. The ⟨ LINE STATION STATEMENT ⟩ may not appear in any ⟨ LINE DEFAULT DEFINITIONS ⟩. A station may be associated with only one line.
5. A ⟨ LINE ADDRESS STATEMENT ⟩ cannot appear in a ⟨ LINE DEFAULT DEFINITION ⟩.

Semantics:

Each line in an NDL/Data Comm system must be described in the LINE Section. Default definitions are provided as a coding aid.

LINE SECTION

AUTOPOLL SIZE Statement

Syntax:

⟨ AUTOPOLL SIZE STATEMENT ⟩ ::= AUTOPOLL = ⟨ INTEGER ⟩

Semantics:

This statement defines the maximum size in characters of an autopoll buffer. This buffer will contain the poll string for the line as supplied by the ⟨ POLL STATEMENT ⟩ in the LINE CONTROL Section.

The maximum value of ⟨ INTEGER ⟩ is 1023. ⟨ integer ⟩ must equal the poll string length multiplied by the number of stations on the line.

LINE ADDRESS Statement

Syntax:

⟨ LINE ADDRESS STATEMENT ⟩ ::= ADDRESS = ⟨ LINE ADDRESS LIST ⟩

⟨ LINE ADDRESS LIST ⟩ ::= ⟨ LINE ADDRESS ⟩
| ⟨ LINE ADDRESS ⟩ , ⟨ LINE ADDRESS LIST ⟩

⟨ LINE ADDRESS ⟩ ::= ⟨ PORT NUMBER ⟩ : ⟨ CHANNEL ADDRESS ⟩ :
⟨ ADAPTER ADDRESS ⟩

⟨ PORT NUMBER ⟩ ::= 0|1|2|3|4|5|6|7

⟨ CHANNEL ADDRESS ⟩ ::= 0|1|2|3|4|5|6|7|8|9|10|11|12|13|14|15

⟨ ADAPTER ADDRESS ⟩ ::= 0|1|2|3|4|5|6|7|8|9|10|11|12|13|14|15

Semantics:

The ⟨ LINE ADDRESS STATEMENT ⟩ is used to specify the port number, the channel number, and the adapter number within the channel to which the line is connected. Two lines may not have the same address.

Example:

ADDRESS = 2 : 0 : 15.

This statement would appear in the ⟨ LINE DEFINITION ⟩ of the line that is connected to channel number zero at adapter number 15 of port number two.

In a switched-line Data Comm environment, there usually is no predetermined hardware connections as is the usual case with leased lines. Several different station configurations can be alternatively connected to several different line adapters.

A “line group” is a set of line addresses possessing common line attributes that are grouped together under one ⟨ LINE IDENTIFIER ⟩. A line group has a station list which comprises all of the stations that can dial in on any of the lines.

It is the user’s responsibility to eliminate inactive stations through the use of the TERMINATE RELEASE STATION Statement in REQUESTS and the System Status Variable, STATION (LINE).

Line groups are not required with the use of switched lines, but are provided in order to allow the more complicated switched-line networks to be easily defined in NDL.

Example:

LINE X:

ADDRESS = 1:0:0, 1:0:1, 1:0:2.

If a station can dial in on only one line, the LINE ADDRESS Statement need contain only one entry.

LINE CONTROL Statement

Syntax:

⟨ LINE CONTROL STATEMENT ⟩ ::= CONTROL = ⟨ CONTROL IDENTIFIER ⟩

Semantics:

This is a required statement in the LINE Section of every NDL program. It associates a LINE CONTROL procedure with this line.

Example:

CONTROL = C1.

In the LINE CONTROL Section there will be a procedure named CONTROL C1. The ⟨ LINE CONTROL STATEMENT ⟩ in the LINE Section associates that control with the line being described.

LINE DEFAULT Statement

Syntax:

⟨ LINE DEFAULT STATEMENT ⟩ ::= DEFAULT = ⟨ LINE DEFAULT IDENTIFIER ⟩

Semantics:

This statement specifies the ⟨ LINE DEFAULT IDENTIFIER ⟩ of a preceding ⟨ LINE DEFAULT DEFINITION ⟩. The attributes in the ⟨ LINE DEFAULT DEFINITION ⟩ are added to those specified in the ⟨ LINE DEFINITION ⟩ that includes the ⟨ LINE DEFAULT STATEMENT ⟩.

Restrictions:

1. If a default attribute conflicts with an explicit attribute in the applicable ⟨ LINE DEFINITION ⟩, the explicit attribute is given precedence.
2. The ⟨ LINE DEFAULT STATEMENT ⟩ may not appear in a ⟨ LINE DEFAULT DEFINITION ⟩. Defaults may not be nested.
3. A ⟨ LINE DEFAULT DEFINITION ⟩ must precede a reference to it in a ⟨ LINE DEFAULT STATEMENT ⟩.

LINE STATION Statement

Syntax:

⟨ LINE STATION STATEMENT ⟩ ::= STATION = ⟨ LIST OF STATIONS ⟩

LINE SECTION

$\langle \text{LIST OF STATIONS} \rangle ::= \langle \text{STATION IDENTIFIER} \rangle$
 $\quad | \langle \text{STATION LIST} \rangle , \langle \text{STATION IDENTIFIER} \rangle$

Semantics:

Each line in the NDL/Data Comm network requires this statement in its $\langle \text{LINE DEFINITION} \rangle$. The statement identifies the stations on the line. $\langle \text{STATION IDENTIFIER} \rangle$ must be defined in the STATION Section of the NDL program.

Restrictions:

1. Lines that are designated as PASSIVE must not have a $\langle \text{LINE STATION STATEMENT} \rangle$ in their definitions.
2. A station may be associated with only one line and may not be listed twice for the same line.

LINE TYPE Statement

Syntax:

$\langle \text{LINE TYPE STATEMENT} \rangle ::= \text{TYPE} = \text{PASSIVE}$

Semantics:

A line can be declared in NDL but kept inactive by stating that TYPE = PASSIVE in the LINE Section. If this optional statement is omitted, the line will be considered to be active and its TYPE will be determined through the hardware.

Example:

```
LINE L1:  
  ADDRESS = 4 : 2 : 1.  
  CONTROL = AUTOPOLCTL.  
  STATION = B1.  
  TYPE = PASSIVE.
```

File Section

Syntax:

⟨ FILE SECTION ⟩ ::= ⟨ FILE LIST ⟩

⟨ FILE LIST ⟩ ::= ⟨ FILE DEFINITION ⟩
 | ⟨ FILE LIST ⟩ ⟨ FILE DEFINITION ⟩
 | ⟨ FILE DEFAULT DEFINITION ⟩ ⟨ FILE LIST ⟩

⟨ FILE DEFINITION ⟩ ::= FILE ⟨ FILE IDENTIFIER ⟩ : ⟨ FILE ATTRIBUTE LIST ⟩

⟨ FILE DEFAULT DEFINITION ⟩ ::= FILE DEFAULT ⟨ FILE DEFAULT IDENTIFIER ⟩
 : ⟨ FILE ATTRIBUTE LIST ⟩

⟨ FILE ATTRIBUTE LIST ⟩ ::= ⟨ FILE ATTRIBUTE STATEMENT ⟩
 | ⟨ FILE ATTRIBUTE LIST ⟩ ⟨ FILE ATTRIBUTE
 STATEMENT ⟩

⟨ FILE ATTRIBUTE STATEMENT ⟩ ::= ⟨ FAMILY STATEMENT ⟩
 | ⟨ FILE DEFAULT STATEMENT ⟩
 | ⟨ RESIDENT STATEMENT ⟩

⟨ FILE IDENTIFIER ⟩ ::= ⟨ IDENTIFIER ⟩

⟨ FILE DEFAULT IDENTIFIER ⟩ ::= ⟨ IDENTIFIER ⟩

Restriction:

The ⟨ FILE DEFAULT STATEMENT ⟩ may appear in any ⟨ FILE DEFINITION ⟩ but not in any ⟨ FILE DEFAULT DEFINITION ⟩. Defaults may not be nested. The ⟨ FAMILY STATEMENT ⟩ may not appear within a ⟨ FILE DEFAULT DEFINITION ⟩.

Semantics:

The FILE Section is used to associate stations that are part of an NDL Data Comm network with the remote files of application programs. File definitions allow sets of stations to be grouped into logical entities called files. File names are used by object programs to identify the station, or set of stations, with which they communicate. For example, the file name (file identifier) is the same name as specified in the VA OF ID clause in the FILE SECTION (DATA DIVISION) of a COBOL program.

FAMILY Statement

Syntax:

⟨ FAMILY STATEMENT ⟩ ::= FAMILY = ⟨ STATION FAMILY ⟩ | FAMILY = ALL

⟨ STATION FAMILY ⟩ ::= ⟨ STATION IDENTIFIER LIST ⟩ | DUMMY

⟨ STATION IDENTIFIER LIST ⟩ ::= ⟨ STATION IDENTIFIER ⟩
 | ⟨ STATION IDENTIFIER ⟩ ,
 ⟨ STATION IDENTIFIER LIST ⟩

Semantics:

The names of stations which are members of the file being described are listed with this statement. A station can be associated with any number of files, and it can also be listed any number of times in any one file.

FILE SECTION

If DUMMY is used, an MCS must assign the stations to be associated with the file at file open time, and the number of stations assigned must equal the number of stations declared in the FILE statement of the application program that opens the file.

ALL will include all stations declared in the file.

FILE DEFAULT Statement

Syntax:

⟨ FILE DEFAULT STATEMENT ⟩ ::= DEFAULT = ⟨ FILE DEFAULT IDENTIFIER ⟩

Semantics:

This statement specifies the ⟨ FILE DEFAULT IDENTIFIER ⟩ of a preceding ⟨ FILE DEFAULT DEFINITION ⟩. The attributes in the ⟨ FILE DEFAULT DEFINITION ⟩ are added to the explicit attributes specified in the ⟨ FILE DEFINITION ⟩ in which this statement appears.

The ⟨ DEFAULT STATEMENT ⟩ in the FILE Section provides compatibility with the DEFAULT statements in other sections; however, its use is limited by the FAMILY = and RESIDENT = attributes of this section.

RESIDENT Statement

Syntax:

⟨ RESIDENT STATEMENT ⟩ ::= RESIDENT ⟨ TYPE ⟩

⟨ TYPE ⟩ ::= DISK
 | CORE

Semantics:

This statement is optional. It can be used to determine if the application program that opened this remote file will be rolled out to disk while waiting for input from this file. The default is CORE when this statement is omitted.

SECTION 4

MESSAGE CONTROL SYSTEM

DESCRIPTION OF MESSAGE CONTROL SYSTEMS

The B 1800/B 1700 data communications subsystem provides an interface between the Network Controller and any executing Message Control System (MCS) on the system. The character-oriented header information which the MCS reads or writes in its communication with the Network Controller, the remote files in its own network, and the MCP basically constitute the MCS interface that is described in this manual. The MCS interface, then, is composed of the various messages required for any queries or changes in the status of remote stations.

This data communications environment requires a Network Controller and a user program which opens a remote file with HEADERS (an MCS). There is no restriction on the number of remote files with HEADERS that can be opened on the system, although a maximum of 20 remote files can be concurrently associated with any one station. In a system composed of several message control systems, a station can be associated with more than one MCS in a hierarchical manner.

In general, the Network Controller handles the line protocol and the MCS, in cooperation with the Network Controller, handles the attachment of remote stations to their respective remote files. Remote file I/O, as is standard on B 1800/B 1700 systems, is controlled by the operating system through a queue file mechanism that is transparent to the user.

An MCS program fulfills some or all of the following data communications needs:

- a. Message switching.
- b. Logical attachment of a station to a remote application program or system of programs.
- c. Network reconfiguration.
- d. Audit and recovery.
- e. Network statistical analysis.
- f. Communication with the MCP.

THE REMOTE FILE INTERFACE

Remote files are the means by which programs use the NDL data communications subsystem to transfer information from remote terminals to user programs (or vice versa). MCS-type remote files are distinguished from ordinary remote files only by the special header that contains additional information about the message.

The MCS interface is enabled by opening a remote file with HEADERS that has an external file name which matches a file declared in the FILE Section of the executing Network Controller. All stations listed in the FAMILY statement for that file are controlled by the MCS. The 50-byte header that precedes data messages allows the MCS to access tallies, toggles, and other information relevant to the originating station of the message. Non-data messages consist only of the variable-length header.

Dummy remote files (those with no stations specified) are allowed providing that the program opening the dummy remote file (without HEADERS) has been zip-executed by an MCS-type program. The MCS is expected to direct the assignment of stations to that file by approving or disapproving the open of any station that was requested to be attached to that file.

MESSAGE CONTROL SYSTEM

SHARING RESOURCES AMONG MULTIPLE MCS FILES

Where multiple Message Control Systems are executing on the same system, there is an established protocol which allows one MCS to attach stations to a remote file of another MCS. The protocol is based upon the concepts of primary and secondary files; these terms refer to the relative responsibilities of two Message Control Systems in the control of a station.

Since users may sign on to a series of MCS-type files, primary files are distinguished from secondary files only by signal characters and by their relative positions in the master list of attachments kept for each station in the remote network. The primary file is the last remote file on the master to which the station is attached and the secondary file is the next-to-last. By default, all interface messages go to the primary file, but if the first character of the message matches the signal character defined for the secondary file (all secondary files must have a signal character), the message goes to the secondary file.

The advantage of this configuration is that a remote station can still communicate with the secondary file even though it has a primary attachment to another remote file.

Secondary files, then, are restricted to remote files opened with HEADERS since they must have signal characters associated with them. Primary files can be either ordinary remote files or remote files opened with HEADERS. In a series of remote attachments done by an individual station, there is a limit of one attachment to a remote file without headers; that remote file must be the primary file. If all of the attachments are to MCS-type remote files, there is a limit of 20 attachments for one remote station.

Primary and secondary file protocol is maintained through a master list that is updated by the Network Controller each time a remote station attaches or detaches itself from an MCS. When a station is attached to a remote file, the file-name is added to the list and a signal character is associated with the primary file. If a third attachment is made, the primary and secondary designations are reconfigured. After redefinition, the original MCS does not have any current responsibilities where the remote station is concerned and is, at this point, inaccessible from that station.

When a remote station is detached or closed, the last entry is deleted from the list and primary and secondary files are reconfigured from the master list. In this way, the original MCS (in a series of three attaches) is maintained on the list and the final close/detach operation must be from the first (original) MCS. Message control systems designed to inhibit the primary-secondary option do so by denying all opens on remote files with headers.

MCS MESSAGES AND REPLIES

General

There are three types of record formats: (1) data, (2) control, and (3) user-defined. The data record format is used in the transfer of information between Network Controllers and remote stations, and in this area the MCS may READ or WRITE the record, depending on the source and destination of the message. In the control record format, the records are defined by name and represent the specific purpose and action, a DETACH-REPLY for example. The user-defined record format is used for communication between an MCS and a user remote file without HEADERS and, as its name indicates, allows the user program to establish the purpose of the communication.

Table 4-1 contains the names of the messages that can be read by an MCS.

Table 4-1

Messages The MCS Can Read

Message Name	Message Type	Message Format
OUTPUT (from a remote file)	00	Data
INPUT (from a station)	01	Data
INPUT-LOGICALACK	02	Data
GOOD-RESULTS-REPLY	05	Data
RECALLED	06	Data
UNPROCESSED-OUTPUT	07	Data
OPEN	10	Open
ATTACH	12	Attach
ATTACH-REPLY	13	Attach
DETACH	14	Detach
DETACH-REPLY	15	Detach
CLOSE	16	Close
STATUS-REPLY	21	Status
CHANGE-REPLY	23	Change
RECALL-REPLY	25	Recall
REMOVE-REPLY	27	Recall
REMOTE-FILE-INFO-REPLY	29	Remote File Info

**MESSAGE CONTROL
SYSTEM**

Table 4-2 contains the names of the messages that can be written by the MCS.

Table 4-2
Messages The MCS Can Write

Message Name	Message Type	Message Format
OUTPUT (to a station)	00	Data
INPUT (to a remote file)	01	Data
LOGICALACK-REPLY	03	Data
OUTPUT-GOOD-RESULTS	04	Data
OPEN-REPLY	11	Open
ATTACH	12	Attach
ATTACH-REPLY	13	Attach
DETACH	14	Detach
STATUS	20	Status
CHANGE	22	Change
RECALL	24	Recall
REMOVE	26	Recall
REMOTE-FILE-INFO	28	Remote File Info

DATA MESSAGES

General

The data messages that are associated with an MCS can be used to read or write, and are explained as follows:

An MCS can read:

- a. An OUTPUT message (type 00) from a remote file whose open the MCS approved with PARTICIPATING set to 1.
- b. An INPUT message (type 01) from:
 1. A primary station when the signal character is not used, or
 2. A secondary station when the signal character designated in the OPEN, ATTACH, or ATTACH-REPLY message is the first character of the message, or
 3. A remote file with headers.
- c. An INPUT-LOGICALACK message (type 02) from the Network Controller. This message is received when a REQUEST executes a TERMINATE LOGICALACK. If the primary file has headers, it receives this message; otherwise, if there is a secondary file, it gets the message.
- d. A GOOD-RESULTS-REPLY message (type 05) from the Network Controller. This message is received upon successful transmission of an OUTPUT message to a station. The GOOD-RESULTS-REPLY message is received only if the GOOD-RESULTS bit in the message or station is set. If the primary remote file has headers, this message is received by the primary file; otherwise, the secondary file receives the message.
- e. A RECALLED message (type 06) from the Network Controller. RECALLED messages follow the RECALL-REPLY message. The number of recalled messages is indicated in the RECALL-REPLY message.
- f. An UNPROCESSED-OUTPUT message (type 07) from the Network Controller. When the Network Controller is DS'ed, it sends output messages to the appropriate controlling remote file before sending an EOF.
- g. A data message with message type greater than or equal to 50 from:
 1. A remote file with headers, or
 2. A remote file whose open was approved by the MCS with USE-REMOTE set.

An MCS can write:

- a. An OUTPUT message (type 00) to any station in its remote file.
- b. An INPUT message (type 01) to any remote file. The destination of the message is indicated in the REMOTE-FILE-NO field.
- c. A LOGICALACK-REPLY message (type 03) to the Network Controller. This message should allow the relevant request to acknowledge receipt of the message by sending an ACK to the station.
- d. A GOOD-RESULTS message (type 04) to the Network Controller. This message acts like an OUTPUT message, except that positive receipt of the message by the station produces a GOOD-RESULTS-REPLY message.
- e. A data message whose message type is greater than or equal to 50. This message will be sent to the REMOTE-FILE-NO specified.

**MESSAGE CONTROL
SYSTEM**

Data Message Format

Table 4-3 illustrates the format of data messages.

Table 4-3
Data Message Format

Field Name	NC-MCS		MCS-NC		MCS-USER		USER-MCS		Field Length PIC
	W	R	W	R	W	R	W	R	
MESSAGE-TYPE	*	*	+	*	+	(*)	(*)	*	99
VARIANT	*	*	*	*	*	-	-		9
LSN	*	*	*	*	*	(*)	(*)	*	999
TEXT-SIZE	*	*	+	*	+	(*)	(*)	*	9(4)
REMOTE-FILE-NO	*	*	*	*	+	-	-		999
TIME	*	*				-	-		9(7)
TRAN-NO	*	*				-	-		999
ERROR	*	*				-	-		XX
TALLYS	*	*	*	*		-	-		9(9)
TOGGLES	*	*	*	*		-	-		9(8)
TERMINAL-TYPE	*	*				-	-		99
FILLER									X(6)
TEXT	*	*	*	*	*	*	*	*	X(#)

The meanings of the abbreviations and symbols used in table 4-3 are as follows:

Abbreviation or Symbol	Meaning
NC	Network Controller/operating system interface
MCS	Remote file with headers
USER	Remote file without headers
R	Read
W	Write

Data Message Format (Cont)

<u>Abbreviation or Symbol</u>	<u>Meaning</u>
*	An asterisk under a “W” implies that it is appropriate for the type of program in that column to initialize that field. An asterisk under an “R” implies that it is appropriate for the type of program in that column to read that field.
+	A plus sign under a “W” implies that it is mandatory for the MCS to initialize that field before sending the message.
9	The numeral “9” implies a numeric EBCDIC character field.
X	The letter “X” implies an EBCDIC character field.

The asterisk enclosed within parentheses in the MCS-USER and USER-MCS columns indicates the three fields of the remote key. The remote key has the following format:

<u>Field Name</u>	<u>Field Length</u>
RSN	9(3)
TEXT-SIZE	9(4)
MESSAGE-TYPE	9(3)

RSN is converted to LSN by the remote file interface.

Semantics of the Data Messages

The semantics of the fields in the message header for data message format are:

MESSAGE-TYPE MESSAGE-TYPE is described in detail in tables 4-1 and 4-2.

VARIANT is as follows:

<u>Value</u>	<u>Description</u>
1	Leave the application program in memory on a remote file read by the application program rather than rolling the program out to disk if the remote file has no messages for the application program.
2	Roll the program out to disk on a remote file read, if no messages are present in the remote file for the application program.
3	Cause the application program to execute its end of file branch.
4	Cause the application program to execute its exception branch. Note: VARIANT will also be set to 4 on a MESSAGE-TYPE of 01 when read by the MCS if a Network Controller request executes the NDL construct TERMINATE ERROR.
5	Retain the text on a GOOD-RESULTS-REPLY message.

MESSAGE CONTROL SYSTEM

LSN	LSN is the logical station number to which any output message to a station is to be sent. It must be set by the MCS or left unchanged (if a message already has a correct LSN) on any message destined for a station from the MCS.
TEXT-SIZE	TEXT-SIZE is the size of the message in terms of characters.
REMOTE-FILE-NO	REMOTE-FILE-NO is the number of the remote file where the message came from or is going to. In order to write to an application program, REMOTE-FILE-NO must be set. REMOTE-FILE-NO is always obtained from the OPEN message.
TIME	The time at which the Network Controller started processing the message.
TRAN-NO	TRAN-NO is the current input transmission number of a message received from a station. An output TRAN-NO may be set to any value by the MCS and that value will be set into the corresponding station table and used in the next Network Controller-station output transmission.
ERROR	ERROR is a 16-bit field formulated, in part, from the RS (result status) field of the I/O descriptor used to perform the I/O operation that resulted in this message.

Fields such as format error and address error are set by the Network Controller during its own checking. The bits of the ERROR fields have the following meanings:

<u>Bit</u>	<u>Meaning</u>
0	Parity error
1	Buffer overflow
2	Memory parity on fetching a character
3	Time out
4	Break
5	End of buffer
6	Loss of data set ready
7	Loss of carrier
8	Address error
9	Translate error
10	Format error
11	Read not ready
12	Exception occurred
13	ACU adapter
14-15	Reserved

TALLYS	TALLYS represent TALLIES 0-2 of the station table.
TOGGLES	TOGGLES represent TOGGLES 0-7 of the station table.
TERMINAL-TYPE	TERMINAL-TYPE represent a field in the terminal table which contains the value specified by the TYPE statement in the TERMINAL Section of the Network Controller definition.
TEXT	The text of the message.

OPEN AND OPEN-REPLY MESSAGES

General

The MCP receives an OPEN message when an application program opens a remote file. A remote file OPEN message is then formulated and passed to the Network Controller, which takes the following actions:

- a. If the file is known, it modifies the message to indicate the appropriate station list; if the file is unknown, the open is disapproved with FILE MISSING.
- b. Verifies that the OPEN message is valid. If not, it disapproves the open.
- c. If none of the logical station numbers in the OPEN message are assigned to an MCS, the Network Controller approves the open and creates a new remote file.
- d. If some of the stations in the OPEN message are assigned to one MCS and rest are unassigned, the Network Controller forwards the OPEN message to the MCS.
- e. If the stations in the OPEN message are assigned to more than one MCS, the Network Controller disapproves the open with FILE LOCKED.
- f. If the program whose open is being processed was zip-executed, the job number of the program that did the zip will be found in PARENT-JOB-NUMBER. In this case, the open is forwarded to the file with headers which belongs to the program with the indicated job number. If no such file exists, then the open is denied.
- g. If the OPEN message is of a dummy file with headers but was not zip-executed by an MCS, the open is approved.
- h. If the OPEN message is of a file without headers and the program was not zip-executed by an MCS, it will be disapproved. This is done because there is no way to attach stations to the remote file after the open.

If the open is passed to an MCS, an OPEN-REPLY message is expected. The MCS must approve or deny the open. The MCS may also modify a number of other fields as shown in the column heading MCS-NC W of table 4-4. If a denial is sent by the MCS, no changes are made, and the denial is forwarded to the operating system which denies the open to the opening program. A signal character, indicated in the OPEN-REPLY message, enables the station to communicate with the MCS. If the OPEN-TYPE is OUTPUT or if PARTICIPATING is set, no changes are made to primary or secondary assignments.

If both PARTICIPATING and HEADERS are set, the open message is disapproved by the Network Controller and a CLOSE message with OPEN-ERROR set is dispatched to the MCS.

When the Network Controller receives an OPEN-REPLY message from an MCS, it rechecks all fields relevant to itself and the operating system. If the MCS made an error in formulating the reply, the approval is changed to denial and a CLOSE message is sent to the MCS with OPEN-ERROR = 1. Otherwise, the new remote file is created and the reply is forwarded to the operating system.

An MCS may approve an open with the HEADERS option set, indicating the open was from another MCS. When that is the case, the primary file is the file whose open was approved and the secondary file is the file which approved that open. All messages whose first character is the designated signal character go to the secondary file. All other messages go to the primary file. Then, if the second MCS approves an open with that station or attaches that station to a third remote file, the first MCS is left out and the second MCS becomes the secondary file.

MESSAGE CONTROL SYSTEM

An example might be a station running under the illustrative MCS. The station first signs on to a special MCS, which retrieves certain types of information from a data base on command and then through the second MCS signs onto an inventory review program. The station operator can review the inventory, enter the signal character, and query the data base through the special-purpose MCS. In order to contact the illustrative MCS, however, the station operator must first sign off from the inventory review program.

A second example illustrates when an MCS is participating. For example, a station running under the illustrative MCS signs onto a special MCS which formats messages according to terminal type, sets up forms, displays data attractively, etc., and from this second MCS signs onto the inventory review program. The second open is approved with PARTICIPATING set to 1. In this case, primary messages are sent to the special MCS and secondary messages still go to the illustrative MCS.

Format of the OPEN and OPEN-REPLY Messages

Table 4-4 illustrates the format of the OPEN and OPEN-REPLY messages. Indications of relevant fields are included.

Table 4-4
OPEN and OPEN-REPLY Message Format

Field Name	NC-MCS		MCS-NC		PIC
	W	R	W	R	
MESSAGE-TYPE	*	*	+	*	99
OPEN-TYPE	*	*			9
OPEN-TIME	*	*			9(7)
PARENT-JOB-NO	*	*			9(7)
PROGRAM-JOB-NO	*	*			9(7)
PROGRAM-NAME	*	*			X(30)
HEADER-OPTION	*	*			9
FILLER					X
USE-REMOTE-KEY	*	*			9
RESIDENT	*	*	*	*	9
USER-REMOTE-FILE-NO	*	*		*	999
SIGNAL-CHAR			*	*	X
APPROVE-DENY			+	*	9
FILLER					X
PARTICIPATING			*	*	9
GOOD-RESULTS			*	*	9
MAX-STATIONS	*	*			999
CURRENT-STATIONS	*	*	*	*	999
LIST-TYPE	*	*			9
FILE-NAME	*	*			X(10)
PROTOCOL-TYPE	*	*			99
SESSION	*	*			9999
STATION-LIST	*	*			(999)#

MESSAGE CONTROL SYSTEM

Semantics of the OPEN and OPEN-REPLY Messages

The semantics of the OPEN and OPEN-REPLY messages are as follows:

MESSAGE-TYPE	MESSAGE-TYPE is the field which identifies this message as an OPEN "10" or an OPEN-REPLY "11". It must be set by the MCS on an OPEN REPLY, and will be set by the Network Controller on an OPEN.								
OPEN-TYPE	OPEN-TYPE indicates the directions of data flow allowed the remote file: <table><thead><tr><th><u>Value</u></th><th><u>Meaning</u></th></tr></thead><tbody><tr><td>1</td><td>Input only</td></tr><tr><td>2</td><td>Output only</td></tr><tr><td>3</td><td>Input/output</td></tr></tbody></table>	<u>Value</u>	<u>Meaning</u>	1	Input only	2	Output only	3	Input/output
<u>Value</u>	<u>Meaning</u>								
1	Input only								
2	Output only								
3	Input/output								
OPEN-TIME	OPEN-TIME is the time which the MCP recognized the file OPEN.								
PARENT-JOB-NO	PARENT-JOB-NO is the job number of the program that zip-executed the program opening the remote file or "0000000".								
PROGRAM-JOB-NO	PROGRAM-JOB-NO is the job number of the program opening the remote file.								
PROGRAM-NAME	PROGRAM-NAME is the name of the program opening the remote file.								
HEADER-OPTION	HEADER-OPTION indicates whether the file is allocated MCS-type headers and functions.								
USE-REMOTE-KEY	USE-REMOTE-KEY indicates whether the file includes the key option. The REMOTE-KEY option allows writes to specific stations, and on reads indicates the station which sent the current message. For files without headers, stations are identified by relative station numbers (RSNs).								
RESIDENT	RESIDENT is a value indicating what to do on a read with no messages present: <table><thead><tr><th><u>Value</u></th><th><u>Meaning</u></th></tr></thead><tbody><tr><td>1</td><td>Keep program in memory.</td></tr><tr><td>2</td><td>Roll program out to disk.</td></tr><tr><td>3</td><td>Provide EOF branch.</td></tr></tbody></table>	<u>Value</u>	<u>Meaning</u>	1	Keep program in memory.	2	Roll program out to disk.	3	Provide EOF branch.
<u>Value</u>	<u>Meaning</u>								
1	Keep program in memory.								
2	Roll program out to disk.								
3	Provide EOF branch.								
USER-REMOTE-FILE-NO	USER-REMOTE-FILE-NO is the logical file number which the Network Controller uses to identify the opening file throughout the remote file interface. The MCS must not change this field.								
SIGNAL-CHAR	SIGNAL-CHAR is used by the Network Controller to identify messages intended for the MCS from a station. Blank implies no signal character.								
APPROVE-DENY	APPROVE-DENY indicates to the operating system whether the OPEN should be approved. APPROVE-DENY = 1 implies OPEN approval.								

MESSAGE CONTROL SYSTEM

PARTICIPATING	PARTICIPATING is a one-bit field that indicates whether the approving MCS will participate in the user program's I/O. It is set by the approving MCS and causes all input from the station and all output from the remote file to be sent to the approving MCS rather than the user program. No changes are made to the primary or secondary files of the stations in a remote file OPEN with PARTICIPATING set to 1.
GOOD-RESULTS	GOOD-RESULTS is set by the MCS to indicate that the Network Controller should return GOOD-RESULTS messages upon successful transmission of data messages to the station.
MAX-STATIONS	MAX-STATIONS indicates the number of stations that can be attached to a given file. It is set by the program attempting to open the remote file as part of the file declaration.
CURRENT-STATIONS	CURRENT-STATIONS represents the number of stations that are in the STATION-LIST to be originally attached to the file. CURRENT-STATIONS equal to "000" indicates a dummy file, which only the approving MCS (or any MCS gaining knowledge of its remote file number) can communicate with, and to which stations could later be attached.
LIST-TYPE	LIST-TYPE indicates the method by which the user program specified its remote file. The value of this field is always 0, which indicates that FILE-NAME is ordered by file name rather than by any other method.
FILE-NAME	FILE-NAME is a field providing the file name given by the user program. This name will match a file name declared in the FILE Section of the NDL program which generated the Network Controller. However, the file name is not a unique file identifier. The stations in a given NDL file may be modified so as to be shared by two remote files or passed on to another remote file with the same name. In future software releases, files may also be designated by station list, so reliance on file names to identify remote files is not recommended.
STATION-LIST	STATION-LIST contains the list of stations (by LSN) included in the remote file. Each LSN occupies 3 characters in the list. Refer to the description of CURRENT-STATIONS for additional information.
PROTOCOL-TYPE	Indicates the type of remote file intercommunication desired by the user program opening the file. The defined values are: 00 - Input messages are type 01. 01 - Input messages are type 50 or greater.
SESSION	The remote session number associated with the user program performing the OPEN. SESSION is equal to 0000 if there is no session association.

MESSAGE CONTROL SYSTEM

Network Controller OPEN Review Criteria

The following conditions are checked when an open is approved by the Network Controller without being sent to an MCS:

- a. There is room in the remote file table as indicated by the MAX FILES statement in the NDL DECLARATION Section.
- b. CURRENT-STATIONS is less than or equal to MAX-STATIONS; if not, it is set to MAX-STATIONS.
- c. For all stations in the list:
 1. The station exists, and
 2. The adapter is present, and
 3. The station meets the following conditions:
 - A. The old primary is null, or
 - B. The old primary does not have headers, and the old secondary is null and the OPEN-TYPE is OUTPUT (2)
- d. The file is not a dummy file without headers.

The following conditions are checked before an OPEN message is forwarded to an MCS:

- a. Adequate space is available in the remote file table as indicated by the MAX FILES statement in the NDL DECLARATION Section.
- b. CURRENT-STATIONS is less than or equal to MAX-STATIONS; if not, it is set to MAX-STATIONS.
- c. If the remote file being opened is a dummy file, it was zip-executed by a program with headers.
- d. For all stations in the list:
 1. The station exists, and
 2. The adapter is present, and
 3. The station meets the following conditions:
 - A. The primary is null (not all stations), or
 - B. The primary is the approving MCS, or
 - C. The primary does not have headers and the secondary is the approving MCS, and
 4. The nesting of OPEN approvals and attaches is less than twenty.

The following conditions are checked before an OPEN-REPLY message is processed and approved for the user program:

- a. The OPEN on this remote file was sent for OPEN approval.

- b. The MCS set APPROVE-DENY to 1, thus indicating approval.
- c. CURRENT-STATIONS is less than or equal to MAX-STATIONS.
- d. HEADERS and PARTICIPATING are not both set.
- e. For all stations in the list:
 - 1. The station exists, and
 - 2. The adapter is present, and
 - 3. The station meets the following conditions:
 - A. The old primary is null, or
 - B. The old primary is the approving MCS, or
 - C. The old primary does not have headers and the old secondary is the approving MCS, and
 - 4. The nesting of OPEN approvals and attaches is less than twenty, and
 - 5. If participating, the primary is the approving MCS.

ATTACH AND ATTACH-REPLY MESSAGES

General

The ATTACH message is a mechanism whereby new stations are assigned to an existing remote file. Whereas the OPEN message allows an initial assignment of stations to a new remote file, the attach protocol adds stations to the file subsequent to the open.

Remote Files Associated With ATTACH

There are three important remote files (not necessarily unique) associated with an ATTACH.

- a. The attach initiator begins the attach process by writing an ATTACH message. Eventually the attach initiator expects to receive an ATTACH-REPLY which either approves or denies the attach. The station list may be modified if the attach is forwarded to another MCS for approval so it may be necessary to review the station list in processing the completed attach.
- b. The attach object is the remote file to which the stations are being attached. If the attach object is not the attach initiator, the OPEN of the attach object must have been approved by the attach initiator. The attach object may or may not have headers. In either case, it receives no indication of the attach within the attach protocol, but may become aware of the attach via the inclusion of new stations in its normal message flow, via a REMOTE-FILE-INFO, or via a user-defined convention.
- c. The Controlling Remote File (CRF) of a station is the file with headers to which the station was most recently attached (or included in an open). If the primary has headers, it is the CRF; otherwise, the secondary file is the CRF, if one exists.

If the CRF of each station in the STATION-LIST is the attach initiator or null, the attach is immediately processed and an ATTACH-REPLY sent back to the attach initiator. The signal character specified in the ATTACH message will direct control messages to the attaching file for all the stations the attach initiator controls.

MESSAGE CONTROL SYSTEM

If the Controlling Remote File (CRF) for the stations in the ATTACH message is a remote file other than the attach initiator, the ATTACH message is forwarded to the CRF and an ATTACH-REPLY message is expected in response. The CRF must approve or deny the attach and may modify the station list of the attach. The attach initiator also specifies the signal character for all stations it controls. The ATTACH-REPLY message is then reviewed by the NC, which either approves the entire attach and processes it or denies it. In either case, the ATTACH-REPLY message is then reviewed by the NC, which either approves the entire attach and processes it or denies it. The ATTACH-REPLY message is then sent on to the attach initiator. If another MCS approves the attach but the NC denied it, a DETACH is also sent to the CRF with ATTACH-REPLY-IN-ERROR set to 1.

If a station is unattached, it may be included in an attach. However, when the attach is processed by the NC, no secondary file is assigned.

If the stations in the STATION-LIST have more than one CRF, the ATTACH is denied by the Network Controller.

Table 4-5 indicates the status of a station before the issuance of an ATTACH and after the receipt of an ATTACH-REPLY (not participating or output only).

The meanings of the words and abbreviations used in table 4-5 are as follows:

<u>Word or Abbreviation</u>	<u>Meaning</u>
p	Primary file
s	Secondary file
self	Attach initiator (headers)
CRF	Controlling Remote File (not self) (headers)
user	Remote file whose open was approved by self (no headers)

If stations are to be attached to a file with PARTICIPATING set or OPEN TYPE output only, no changes will be made to primary or secondary assignments.

Table 4-5

Station Status

Current Status	Status if ATTACH to Self Approved	Status if ATTACH to Other File Approved
<p>UNATTACHED</p> <p>primary = null secondary = null</p>	<p>primary = self secondary = null SIGNAL-CHAR is ignored ATTACH-REPLY from the Network Controller</p>	<p>primary = other secondary = null SIGNAL-CHAR is ignored ATTACH-REPLY from the Network Controller</p>
<p>ATTACHED to self</p> <p>primary = self secondary = null</p>	<p>primary = self secondary = null SIGNAL-CHAR is ignored ATTACH-REPLY from the Network Controller</p>	<p>primary = other secondary = self SIGNAL-CHAR from the ATTACH message is active ATTACH-REPLY from the Network Controller</p>
<p>ATTACHED to CRF</p> <p>primary = CRF secondary = variable</p>	<p>primary = self secondary = CRF SIGNAL-CHAR from the ATTACH-REPLY message is active ATTACH-REPLY from the CRF</p>	<p>primary = other secondary = CRF SIGNAL-CHAR from the ATTACH-REPLY message is active ATTACH-REPLY from the CRF</p>
<p>ATTACHED to file without HEADERS</p> <p>primary = user secondary = null</p>	<p>ATTACH denied by Network Controller ATTACH-REPLY from the Network Controller</p>	<p>ATTACH denied by Network Controller ATTACH-REPLY from the Network Controller</p>
<p>primary = user secondary = self</p>	<p>primary = user secondary = self SIGNAL-CHAR is ignored ATTACH-REPLY from the Network Controller</p>	<p>primary = other secondary = self SIGNAL-CHAR from the ATTACH message is active ATTACH-REPLY from the Network Controller</p>
<p>primary = user secondary = CRF</p>	<p>primary = self secondary = CRF SIGNAL-CHAR from the ATTACH-REPLY message is active ATTACH-REPLY from the CRF</p>	<p>primary = other secondary = self SIGNAL-CHAR from the ATTACH-REPLY message is active ATTACH-REPLY from the CRF</p>

**MESSAGE CONTROL
SYSTEM**

Format of the ATTACH Message

Table 4-6 shows the format of the ATTACH message.

Table 4-6
Format of the ATTACH Message

Field Name	MCS-NC		NC-CRF		PIC
	W	R	W	R	
MESSAGE-TYPE	+	*		*	99
USER-REMOTE-FILE-NO	+	*		*	999
ATTACHING-REMOTE-FILE-NO			*	*	999
SIGNAL-CHAR	*	*			X
APPROVE-DENY					9
DENIAL-REASON					9
PROGRAM-JOB-NO			*	*	9(7)
ATTACH-TIME			*	*	9(7)
CURRENT-STATIONS	+	*		*	999
FILLER					X(6)
LSN-LIST	+	*		*	(999)#

MCS - attach initiator

CRF - controlling remote file

Format of the ATTACH-REPLY Message

Table 4-7 shows the format of the ATTACH-REPLY message.

Table 4-7

Format of the ATTACH-REPLY Message

Field Name	CRF-NO		NC-MCS		PIC
	W	R	W	R	
MESSAGE-TYPE	+	*	*	*	99
USER-REMOTE-FILE-NO		*		*	999
ATTACHING-REMOTE-FILE-NO		*	*	*	999
SIGNAL-CHAR	*	*		*	X
APPROVE-DENY	+	*	*	*	9
DENIAL-REASON			*	*	9
PROGRAM-JOB-NO			*	*	9(7)
ATTACH-TIME			*	*	9(7)
CURRENT-STATIONS	*	*		*	999
FILLER					X(6)
LSN-LIST	*	*		*	(999)#

MESSAGE CONTROL SYSTEM

Semantics of the ATTACH and ATTACH-REPLY Messages

The semantics of the ATTACH and ATTACH-REPLY messages are as follows:

MESSAGE-TYPE MESSAGE-TYPE is the field which identifies this message as an ATTACH "12" or ATTACH-REPLY "13". It may be set by either the MCS or the Network Controller on both ATTACH and ATTACH-REPLY depending on where the message originates.

USER-REMOTE-FILE-NO USER-REMOTE-FILE-NO is the file number assigned to the remote file to which the MCS wishes to attach a station(s).

ATTACHING-REMOTE-FILE-NO ATTACHING-REMOTE-FILE-NO is the file number of the MCS originating the ATTACH or ATTACH-REPLY, and is set by the Network Controller before forwarding the message to another MCS.

SIGNAL-CHAR SIGNAL-CHAR is set by the CRF which currently controls the station. It is used by the Network Controller to identify messages intended for the secondary file of a station. Blank implies no signal character. For previously unattached stations and stations whose CRF is attaching the station to itself, the signal character is ignored.

DENIAL-REASON DENIAL-REASON is set by the Network Controller when an attach is denied. It may have the following values:

<u>Value</u>	<u>Reason</u>
1	The remote file does not exist.
2	The file is locked.
3	The adapter is missing.
4	The CRF which currently owns the station denied the attach.
5	Reserved.
6	There is an invalid LSN in LSN-LIST.
7	There are too many MCSs for one of the stations (the nesting of OPENS and ATTACHES is greater than twenty).
8	There is an error in the ATTACH-REPLY message format.
9	There are too many stations in the file.

PROGRAM-JOB-NO PROGRAM-JOB-NO on an ATTACH is the job number of the owner of the attaching remote file. On an ATTACH-REPLY this is the job number of the CRF which currently owns the station. If this field corresponds to ones own file upon receiving an ATTACH-REPLY, the ATTACH was approved by the Network Controller only.

ATTACH-TIME ATTACH-TIME is the time at which the MCP detected the ATTACH.

CURRENT-STATIONS CURRENT-STATIONS represents the number of stations to attach (in an ATTACH message) or that were attached (in an ATTACH-REPLY message).

LSN-LIST LSN is a list, by logical station number of stations to be attached (in an ATTACH message) or that were attached (in an ATTACH-REPLY message).

Network Controller ATTACH Review Criteria

The following conditions are checked before an ATTACH message is approved by the Network Controller, without the ATTACH message being sent to another MCS.

- a. The remote file exists (the field USER-REMOTE-FILE-NO is a valid file number).
- b. The attach initiator approved the open of the remote file or the attach initiator is the user remote file.
- c. The value of CURRENT-STATIONS plus the number of stations already attached to the MCS is less than or equal to the value of MAX-STATIONS for that file.
- d. For all the stations in the station list:
 1. The stations exists, and
 2. The adapter is present, and
 3. The station meets the folloing:
 - A. The old primary is null, or
 - B. The old primary is not the attach initiator, or
 - C. The old primary does not have headers and the old secondary is the attach initiator.
 4. The nesting of open approvals and attaches is less than twenty.
 5. If the file is indicated as participating, the primary is the attach initiator.

The following conditions are checked before an attach is forwarded to the controlling remote file (not the attach initiator) for approval:

- a. The remote file exists (the field USER-REMOTE-FILE-NO is a valid file number).
- b. The attach initiator approved the open of the remote file or the attach initiator is the user remote file.
- c. The value of CURRENT-STATIONS plus the number of stations already attached to the file is less than or equal to the value of MAX-STATIONS for that file.
- d. For all the stations in the station list:
 1. The stations exists
 2. The adapter is present
 3. The station meets the following:
 - A. The old primary is null, or

MESSAGE CONTROL SYSTEM

- B. The old primary is the controlling remote file, or
 - C. The old primary does not have headers and the old secondary is the controlling remote file.
4. The nesting of open approvals and attaches is less than twenty.

The following conditions are checked before an ATTACH REPLY message is processed and approved for the attach initiator:

- a. The remote file exists (the field USER-REMOTE-FILE-NO is a valid file number).
- b. The ATTACH message has been forwarded to the appropriate MCS for approval.
- c. That MCS set the field APPROVE-DENY to a one to indicate approval.
- d. The value of CURRENT-STATIONS plus the number of stations already attached to the file is less than or equal to the value of MAX-STATIONS for that file
- e. For all the stations in the station list:
 - 1. The stations exist and
 - 2. The adapter is present and
 - 3. The station meets the following:
 - A. The old primary is null, or
 - B. The old primary is the controlling remote file, or
 - C. The old primary does not have headers and the old secondary is the controlling remote file.
 - 4. The nesting of open approvals and attaches is less than twenty.

DETACH AND DETACH-REPLY MESSAGES

General

The DETACH is related to the CLOSE in the same way that ATTACH is related to the OPEN message. DETACH is provided for negating the effect of an ATTACH, removing stations from the station list of a remote file, and returning them to the files to which they were previously attached.

If the USER-REMOTE-FILE-NO is valid, the DETACH will be processed, station-by-station, according to the following criteria:

- a. A file may detach a station from itself. If there is a CRF which approved the OPEN of that file, it is notified by the DETACH message which is forwarded by the Network Controller. This indicates to the receiving file that it is now the primary file again.
- b. A CRF may detach a station from a file whose ATTACH or OPEN it approved.
- c. When a station is detached from the remote file specified, it is also detached from all files to which it had subsequently become attached.

- d. When an ATTACH-REPLY message has an error, a DETACH message is sent to the CRF with the ATTACH-REPLY-IN-ERROR field set.

There are three different types of DETACH messages:

- a. The first type of DETACH message is sent by a remote file with headers to detach a station from its file or a directly subordinate file. The response to the first type of DETACH message is a DETACH-REPLY message from the Network Controller, with the LSN-LIST indicating the stations actually detached.
- b. The second type of DETACH message is sent by the Network Controller to notify an MCS that it now controls a list of stations. No DETACH-REPLY message is necessary.
- c. The third type of DETACH message is sent by the Network Controller to notify a CRF writing an ATTACH-REPLY message that the ATTACH-REPLY message had an error and the attach did not get processed. No DETACH-REPLY message is necessary.

These three types of DETACH messages are answered by a DETACH-REPLY message from the Network Controller, and the LSN LIST indicates which stations are actually detached.

Format of the DETACH AND DETACH-REPLY Messages

Table 4-8 shows the format of the DETACH message.

Table 4-8. Format of the DETACH Message

Field Name	MCS-NC		NC-MCS		PIC
	W	R	W	R	
MESSAGE-TYPE	+	*	*	*	99
USER-REMOTE-FILE-NO	+	*	*	*	999
ATTACH-REPLY-IN-ERROR			*	*	9
CURRENT-STATIONS	+	*	*	*	999
FILLER					X(6)
LSN-LIST	+	*	*	*	(999)#

MESSAGE CONTROL SYSTEM

Semantics of the DETACH and DETACH-REPLY Messages

The semantics of the DETACH and DETACH-REPLY messages are as follows:

MESSAGE-TYPE	MESSAGE-TYPE is the field which identifies this message as a DETACH "14" or DETACH-REPLY "15". It may be set by either the MCS or the Network Controller on DETACH and by the Network Controller on DETACH-REPLY.
USER-REMOTE-FILE-NO	USER-REMOTE-FILE-NO is the file number of the file from which the MCS wishes to detach stations.
ATTACH-REPLY-IN-ERROR	ATTACH-REPLY-IN-ERROR is set if an ATTACH-REPLY message was incorrectly formatted.
CURRENT-STATIONS	CURRENT-STATIONS represent the number of stations to detach (in a DETACH message) or the number of stations which were detached (in a DETACH-REPLY message).
LSN-LIST	LSN-LIST is a list by logical station number of the stations to be detached (in the DETACH Message) or that were detached (in the DETACH-REPLY message).

CLOSE MESSAGE

General

The CLOSE message is provided as a means of negating a previous OPEN message. CLOSE messages originate from the operating system and are passed to the remote file which approved the file open. Also, a close may be initiated by the Network Controller if an OPEN-REPLY message approving an open was in error (Refer to the OPEN message). The CLOSE message requires no reply.

If an MCS closes its file, the files whose opens it approved are closed, and they receive no new messages. An end-of-file indication is queued after the last currently-queued message. The stations are relegated to the primary-secondary configuration which existed before the closed file was opened. One exception to the above rule is when an MCS participates in user program I/O, in which case a close on the remote files does not alter the primary and secondary files for the station.

Format of the CLOSE Message

Table 4-9 shows the format of the CLOSE message.

Table 4-9. Format of the CLOSE Message.

Field Name	NC-MCS		PIC
	W	R	
MESSAGE-TYPE	*	*	99
CLOSE-TIME	*	*	9(7)
PROGRAM-JOB-NO	*	*	9(7)
USER-REMOTE-FILE-NO	*	*	999
OPEN-ERROR	*	*	9
CURRENT-STATIONS	*	*	999
FILLER			X(6)
LSN-LIST	*	*	(999)#

Semantics of the CLOSE Message

The semantics of the CLOSE Message are as follows:

- MESSAGE-TYPE** MESSAGE-TYPE is the field which identifies this message as a CLOSE “16”. It is set by the Network Controller.
- CLOSE-TIME** CLOSE-TIME is the time at which the MCP receives the user programs CLOSE.
- USER-REMOTE FILE-NO** USER-REMOTE-FILE-NO is the file number of the file which is being closed.
- OPEN-ERROR** OPEN-ERROR is set to 1 if the OPEN-REPLY message was in error.
- CURRENT-STATIONS** CURRENT-STATIONS represent the number of stations currently attached to the file.
- LSN-LIST** LSN-LIST is a list by logical station number of all stations currently attached to the file.

STATUS AND STATUS-REPLY MESSAGES

General

Status is applicable only to stations.

**MESSAGE CONTROL
SYSTEM**

Format of the STATUS and STATUS-REPLY Messages

Table 4-10 shows the format of the STATUS and STATUS-REPLY messages.

Table 4-10. Format of the STATUS and STATUS-REPLY Messages

Field Name	MCS-NC		NC-MCS		PIC
	W	R	W	R	
MESSAGE-TYPE	+	*	*	*	99
LSN	+	*		*	999
REQUESTING-LSN	*			*	999
STATUS-ERROR	-	-	*	*	9
STATION-NAME	-	-	*	*	X(10)
STATION-READY	-	-	*	*	9
STATION-ENABLED	-	-	*	*	9
STATION-MYUSE	-	-	*	*	9
STATION-TERMINAL-TYPE	-	-	*	*	99
STATION-BUFFERSIZE	-	-	*	*	9(5)
STATION-TRAN-NO-SIZE	-	-	*	*	9
STATION-TRAN-RECEIVE	-	-	*	*	XXX
STATION-TRAN-TRANSMIT	-	-	*	*	XXX
STATION-RECEIVE-ADDR-SIZE	-	-	*	*	99
STATION-TRANSMIT-ADDR-SIZE	-	-	*	*	99
STATION-ADDR-RECEIVE	-	-	*	*	X(20)
STATION-ADDR-TRANSMIT	-	-	*	*	X(20)
STATION-MAX-RETRIES	-	-	*	*	999
STATION-PRIORITY-RECEIVE	-	-	*	*	999
STATION-PRIORITY-TRANSMIT	-	-	*	*	999
MESSAGE-COUNT	-	-	*	*	9(4)
DIAGNOSTIC-REQ-ON	-	-	*	*	9
LOGICALACK-ON	-	-	*	*	9
GOOD-RESULTS-ON	-	-	*	*	9

Table 4-10. Format of the STATUS and STATUS-REPLY Messages (Cont)

Field Name	MCS-NC W R	NC-MCS W R	PIC
STATION-TALLIES	- -	* *	9(9)
STATION-TOGGLES	- -	* *	9(8)
STATION-REMOTE-FILE-NO	- -	* *	999
REMOTE-FILE-HAS-HEADERS	- -	* *	9
STATION-PHONE	- -	* *	X(20)
STATION-VALID	- -	* *	9
STATION-LINE-NO	- -	* *	99
STATION-SECONDARY-FILE-NO	- -	* *	999
FILLER			X(10)
LINE-COUNT	- -	* *	99
LINE-INFO	- -	* *	(999)#
LINE #	99		
ACU #	9		

Semantics of the STATUS and STATUS-REPLY Messages

The semantics of the STATUS and STATUS-REPLY messages are as follows:

- MESSAGE-TYPE MESSAGE-TYPE is the field which identifies this message as a STATUS “20” or a STATUS-REPLY “21”. It must be set by the MCS on a STATUS and will be set by the Network Controller on a STATUS-REPLY.

- LSN LSN represents the logical station number for which the status is requested.

- REQUESTING-LSN REQUESTING-LSN is available to identify the station making the status request, but is not necessary.

- STATUS-ERROR STATUS-ERROR is set to 1 if the LSN field represents a station which is invalid; otherwise, it is set to 0.

- STATION-NAME STATION-NAME is the name of the station as defined in the Network Controller specifications.

- STATION-READY STATION-READY is a field in the station table which is set to 1 if the station is currently capable of receiving output messages; otherwise, it is set to 0.

- STATION-ENABLED STATION-ENABLED is a field in the station table which is set to 1 if the Network Controller is currently capable of receiving messages (polling) from the station; otherwise, it is set to 0.

MESSAGE CONTROL SYSTEM

STATION-MYUSE	STATION-MYUSE is a field in the station table which can be set to any one of three values to indicate the following:								
	<table border="0"> <thead> <tr> <th><u>Value</u></th> <th><u>Meaning</u></th> </tr> </thead> <tbody> <tr> <td>1</td> <td>The station is to be used for input only.</td> </tr> <tr> <td>2</td> <td>The station is to be used for output only.</td> </tr> <tr> <td>3</td> <td>The station can be used for both input and output.</td> </tr> </tbody> </table>	<u>Value</u>	<u>Meaning</u>	1	The station is to be used for input only.	2	The station is to be used for output only.	3	The station can be used for both input and output.
<u>Value</u>	<u>Meaning</u>								
1	The station is to be used for input only.								
2	The station is to be used for output only.								
3	The station can be used for both input and output.								
STATION-TERMINAL-TYPE	STATION-TERMINAL-TYPE is a field in the terminal table which is set to the same value as the TYPE statement in the terminal section of the Network Controller definition.								
STATION-BUFFER-SIZE	STATION-BUFFER-SIZE is a field in the terminal table which indicates the maximum size of the buffer which has been allocated for this station to use.								
STATION-TRAN-NO-SIZE	STATION-TRAN-NO-SIZE is the size of the transmission number field in a message header. STATION-TRAN-NO-SIZE is a field in the terminal table.								
STATION-TRAN-RECEIVE	STATION-TRAN-RECEIVE is a field in the station table which contains the current input transmission number.								
STATION-TRAN-TRANSMIT	STATION-TRAN-TRANSMIT is a field in the station table which contains the current output transmission number.								
STATION-RECEIVE-ADDR-SIZE	STATION-RECEIVE-ADDR-SIZE is a field in the terminal table which represents the size of the address in the message header on input.								
STATION-TRANSMIT-ADDR-SIZE	STATION-TRANSMIT-ADDR-SIZE is a field in the terminal table which represents the size in the message header on output.								
STATION-ADDR-RECEIVE	STATION-ADDR-RECEIVE is a field in the station table which contains the input address (poll) of the station.								
STATION-ADDR-TRANSMIT	STATION-ADDR-TRANSMIT is a field in the station table which contains the output (select) address of the station.								
STATION-MAX-RETRIES	STATION-MAX-RETRIES is a field in the station table which represents the number of time a transmission will be repeated in case of an error.								
STATION-PRIORITY-RECEIVE	STATION-PRIORITY-RECEIVE is a field in the station table and represents the value assigned to the input part of the FREQUENCY statement in the Network Controller definition.								
STATION-PRIORITY-TRANSMIT	STATION-PRIORITY-TRANSMIT is a field in the station table and represents the value assigned to the output part of the FREQUENCY statement in Network Controller definition.								
MESSAGE-COUNT	MESSAGE-COUNT represents the number of messages queued for this station.								
DIAGNOSTIC-REQ-ON	DIAGNOSTIC-REQ-ON is set to 1 if the diagnostic request has been invoked by a CHANGE message; otherwise, this field is zero.								
LOGICALACK-ON	LOGICALACK-ON represents whether LOGICALACK is set in the STATION section for this station.								

MESSAGE CONTROL SYSTEM

GOOD-RESULTS-ON	GOOD-RESULTS-ON is set to a 1 when the GOOD-RESULTS-FIELD is set on an OPEN-REPLY approve message; otherwise, this field is 0.
STATION-TALLIES	STATION-TALLIES represents fields in the station table which contain station tallies 0-2.
STATION-TOGGLES	STATION-TOGGLES represent fields in the station table which contain station-toggles 0-7.
STATION-REMOTE- FILE-NO	STATION-REMOTE-FILE-NO represents the number of the remote file to which the station is attached.
REMOTE-FILE-HAS- HEADERS	REMOTE-FILE-HAS-HEADERS is set to a 1 if the remote file whose number is defined in the STATION-REMOTE-FILE-NO field is an MCS; otherwise, the field is 0.
STATION-PHONE	The telephone number assignment for the station.
STATION-VALID	A field in the STATION table which is set to 1 if the station is valid; otherwise, it is set to 0.
STATION-LINE-NO	The current line assignment for the station.
STATION-SECONDARY- FILE-NO	The remote file number of the station's secondary file. If the remote file number is 000, there is no secondary file.
LINE-COUNT	The number of lines on which the station is defined.
LINE-INFO	LINE-COUNT number of 3-character fields describing the line number (LINE) and its associated DIALOUT status (ACU).

**MESSAGE CONTROL
SYSTEM**

CHANGE AND CHANGE-REPLY MESSAGES

General

The CHANGE message is used to alter certain Network Controller attributes which are described in detail in the following pages.

Format of the CHANGE and CHANGE-REPLY Messages

Table 4-11 shows the format of the CHANGE and CHANGE-REPLY messages.

Table 4-11. Format of the CHANGE and CHANGE-REPLY Messages.

Field Name	MCS-NC		NC-MCS		PIC
	W	R	W	R	
MESSAGE-TYPE	+	*	*	*	99
LSN	+	*		*	999
REQUESTING-LSN	*			*	XXX
CHANGE-TYPE	+	*		*	99
CHANGE-RESULT			*	*	9
CHANGE-VALUE	+	*		*	X(20) or XXX or 999 or 9

Semantics of the CHANGE and CHANGE-REPLY Messages

The semantics of the CHANGE and CHANGE-REPLY messages are as follows:

MESSAGE-TYPE MESSAGE-TYPE is the field which identifies this message as a CHANGE “22” or a CHANGE-REPLY “23”. It must be set by the MCS on a CHANGE and will be set by the Network Controller on a CHANGE-REPLY.

LSN LSN is the logical station number of the station to be changed.

REQUESTING-LSN REQUESTING-LSN is the logical station number of the station from which the CHANGE was originated. This field is not used by the Network Controller.

CHANGE-TYPE CHANGE-TYPE indicates the field to be changed. The values are:

<u>CHANGE-TYPE</u>	<u>Field To Be Changed</u>	<u>Format</u>
00	TRAN (RECEIVE)	XXX
01	TRAN (TRANSMIT)	XXX
02	ADDRESS (RECEIVE)	X(20)
03	ADDRESS (TRANSMIT)	X(20)

**MESSAGE CONTROL
SYSTEM**

<u>CHANGE-TYPE</u>	<u>Field To Be Changed</u>	<u>Format</u>
04	FREQUENCY (RECEIVE)	999
05	FREQUENCY (TRANSMIT)	999
06	MAX-RETRY	999
07	ENABLED	9
08	READY	9
09	DIAGNOSTIC-ON	9
10	LOGICALACK-ON	9
11	GOOD-RESULTS-ON	9
12	STATION-PHONE	X(20)

CHANGE-RESULT CHANGE-RESULT returns a 1 if the CHANGE was correct. A 0 indicates that the LSN field was invalid. A 2 indicates that the CHANGE-TYPE was invalid.

CHANGE-VALUE CHANGE-VALUE is the fields new value in left-justified character format.

RECALL, RECALL-REPLY, REMOVE, AND REMOVE-REPLY MESSAGES

General

The RECALL message is provided for removing any number of messages from the top of a station's output queue, marking them as recalled messages, and sending them, prefixed by a RECALL-REPLY message, to the MCS. The RECALL-REPLY message contains the number of messages to follow.

The REMOVE message is provided for removing any number of messages from the top of a station's queue. The Network Controller always responds with a REMOVE-REPLY message indicating the number of messages actually removed.

NOTE

When using RECALL and REMOVE, it is best to make the station NOT READY first; otherwise, the first message may or may not be included, depending on whether the station is currently being processed.

MESSAGE CONTROL SYSTEM

Format of the RECALL, RECALL-REPLY, REMOVE, and REMOVE-REPLY Messages

Table 4-12 shows the format of the RECALL, RECALL-REPLY, REMOVE, and REMOVE-REPLY messages.

Table 4-12. Format of the RECALL, RECALL-REPLY, REMOVE, and REMOVE-REPLY Messages

Field	MCS-NC		NC-MCS		PIC
	W	R	W	R	
MESSAGE-TYPE	+	*	*	*	99
LSN	*	*		*	999
REQUESTING-LSN	*			*	999
MESSAGE-COUNT	+	*	*	*	9(4)
ERROR			*	*	9

Semantics of the RECALL, REMOVE, RECALL-REPLY, and REMOVE-REPLY Messages.

The semantics of the RECALL, REMOVE, RECALL-REPLY and REMOVE-REPLY messages are as follows:

MESSAGE-TYPE	MESSAGE-TYPE is the field which identifies this message as a RECALL “24”, REMOVE “26”, RECALL-REPLY “25”, or REMOVE-REPLY “27”. It must be set by the MCS on a RECALL or REMOVE and will be set by the Network Controller on a RECALL-REPLY, and REMOVE-REPLY.
LSN	LSN is the logical station number of the station from whose output queue the messages are to be recalled or removed.
REQUESTING-LSN	REQUESTING-LSN is the logical station number of the station initiating the recall or remove.
MESSAGE-COUNT	MESSAGE-COUNT is the number of messages to be recalled or removed (RECALL or REMOVE) and in the reply (RECALL-REPLY or REMOVE-REPLY), the number of messages actually recalled or removed. In the RECALL or REMOVE message, setting MESSAGE COUNT to 999 will cause all messages to be recalled or removed.
ERROR	ERROR is set to 1 if the RECALL or REMOVE message is improperly formulated.

REMOTE-FILE-INFO AND REMOTE-FILE-INFO-REPLY MESSAGES

General

An MCS-type program can control a set of stations by opening a remote file with the HEADERS option. In order to provide necessary information about the file just acquired, the REMOTE-FILE-INFO and REMOTE-FILE-INFO-REPLY message protocol is provided.

Format of the REMOTE-FILE-INFO and REMOTE-FILE-INFO-REPLY Messages

Table 4-13 shows the format of the REMOTE-FILE-INFO and REMOTE-FILE-INFO-REPLY messages.

Table 4-13. Format of the REMOTE-FILE-INFO and REMOTE-FILE-INFO-REPLY Messages

Field Name	MCS-NC		NC-MCS		PIC
	W	R	W	R	
MESSAGE-TYPE	+	*	*	*	99
JOB-NO			*	*	9(7)
TIME			*	*	9(7)
REMOTE-FILE-NO	+(X)	*	*	*	999
OUTPUT-MESSAGES-QUEUED			*	*	9(4)
INPUT-MESSAGES-QUEUED			*	*	9(4)
CURRENT-STATIONS			*	*	999
OTHER-RF-REQUEST	*	*			9
OTHER-RF-ERROR			*	*	9
OPEN-APPROVER-RF-NO			*	*	99
FILLER					99
LSN-LIST			*	*	(999)#
OUTPUT-QUEUED-LIST			*	*	(999)#

(*) Mandatory only if OTHER-RF-REQUEST is a 1.

Semantics of the REMOTE-FILE-INFO and REMOTE-FILE-INFO-REPLY Messages

The semantics of the REMOTE-FILE-INFO and REMOTE-FILE-INFO-REPLY messages are as follows:

- MESSAGE-TYPE** MESSAGE-TYPE is a field which identifies this message as a REMOTE-FILE-INFO “28” or a REMOTE-FILE-INFO-REPLY “29”. This is the only field which must be set for a REMOTE-FILE-INFO message. It is set by the Network Controller on a REMOTE-FILE-INFO-REPLY message.
- JOB-NUMBER** JOB-NUMBER is the job number of the job making this request.
- TIME** TIME is the time the reply was sent.
- REMOTE-FILE-NO** The number of the remote file of this program if OTHER-RF-REQUEST is 0. It is the number of the target remote file if OTHER-RF-REQUEST is 1.
- OUTPUT-MESSAGES-QUEUED** OUTPUT-MESSAGES-QUEUED represents the total number of messages queued for all stations attached to this file.
- CURRENT-STATIONS** CURRENT-STATIONS represent the number of stations currently attached to the remote file.

MESSAGE CONTROL SYSTEM

- OTHER-RF-REQUEST Is zero if the request is for the writing MCS's file. It is a one if REMOTE-FILE-NO contains the file number about which information is requested.
- OTHER-RF-ERROR Is one in the REMOTE-FILE-INFO-REPLY message if the requestor set OTHER-RF-REQUEST to one and the REMOTE-FILE-NO supplied was non-existent.
- OPEN-APPROVER-RF-NO The remote file number of the MCS which approved the requestor's open.
- LSN-LIST LSN-LIST has an entry by LSN for every current station.
- OUTPUT-QUEUED-LIST OUTPUT-QUEUED-LIST has the number of output messages queued for each of the current stations. The total should agree with OUTPUT-MESSAGES-QUEUED.

USER MESSAGES

An executing MCS can interface with a user remote file without headers through the record format shown in table 4-14.

Table 4-14. User-Defined Message Format

Field Name	MCS-USER		USER-MCS		Field-Length PIC
	W	R	W	R	
MESSAGE.TYPE	+	*	*	*	99
FILLER	*	-	-		9
LSN	*	*	*	*	999
TEXT.SIZE	+	*	*	*	9(4)
REMOTE.FILE.NO	+	*	*		999
TEXT	*	*	*	*	X(TEXT.SIZE)

The semantics of the fields of the USER-DEFINED message record are:

- MESSAGE.TYPE Established by the user program and must be a number greater than 49 and less than 100.
- LSN The logical station number to which the data belongs.
- TEXT.SIZE The number of characters in the text field.
- REMOTE.FILE.NO The number of the remote file where the message came from or is going to.
- TEXT The character string which is ordinarily displayed on the remote screen or the local SPO.

SECTION 5

USING THE NDL COMPILER

COMPILER INPUT AND OUTPUT FILES

Input

The source-language input to the B 1800/B 1700 NDL Compiler is handled by several input files. The primary source input file is a card file with the internal name CARDS. This file is required for a compilation and is normally a card reader file, but may be label-equated to another file assigned to a different hardware device.

The secondary source input file to the compiler is a disk file with the internal name SOURCE. This file is optional and is normally a serial disk file, although it may be label-equated to another file assigned to a different device. This file is accepted as input to the compiler when the \$ MERGE compiler option is set. The card images in this file (SOURCE) are merged with the card images in the primary input file (CARDS) according to sequence number.

Output

The NDL Compiler processes the input data and produces several output files, one of which is the line printer listing of the source program whose internal name is LINE. The name of the user's Network Controller and the type of compile is printed on the first line of the printer listing.

The code segment dictionaries for the REQUEST and CONTROL sections are printed at the end of the printer listing of the NDL-compiled Network Controller. Segment zero contains a CASE statement on REQUEST or CONTROL index. The CASE calls the actual REQUEST or CONTROL. Segments 1 through n are the compiled REQUESTS and CONTROLS, with one segment per REQUEST or CONTROL, in the order in which they appeared in the NDL source.

A disk file with the internal name NIF is the Network Information File.

The object code file of the user's Network Controller (Data Communications Handler) is a disk file whose internal name is OBJECT, and whose external name is chosen by the user.

When the MERGE option is used to produce a new source code file, the resulting output file will have internal and external name NEWSOURCE. The \$ NEW option must be included if a new source code file is wanted.

NDL CONTROL CARDS

To compile the user's NDL source code with the B 1800/B 1700 NDL Compiler, the following control cards must be used.

```
? COMPILE USERDCH WITH NDL TO LIBRARY
? DATA CARDS
  _____
  _____
  _____
  _____
  _____
? END
```

} NDL SOURCE CODE

In this case the name USERDCH was chosen for the name of the user's Network Controller.

The question mark (?) indicates an invalid punch that must be in column 1 of the control card.

When ?COMPILE FOR SYNTAX WITH NDL is used, the only NDL Compiler output is a printer listing of the source code with any syntax errors.

Figure 5-1 is a pictorial representation of a source program card deck to be processed by the NDL Compiler.

NDL SOURCE CARD FORMAT

Source Code

The format of the source records that are input to the B 1800/B 1700 NDL Compiler are contained in card columns 1-72 inclusive. Card columns 1-72 inclusive are free-form and may contain source code or comments.

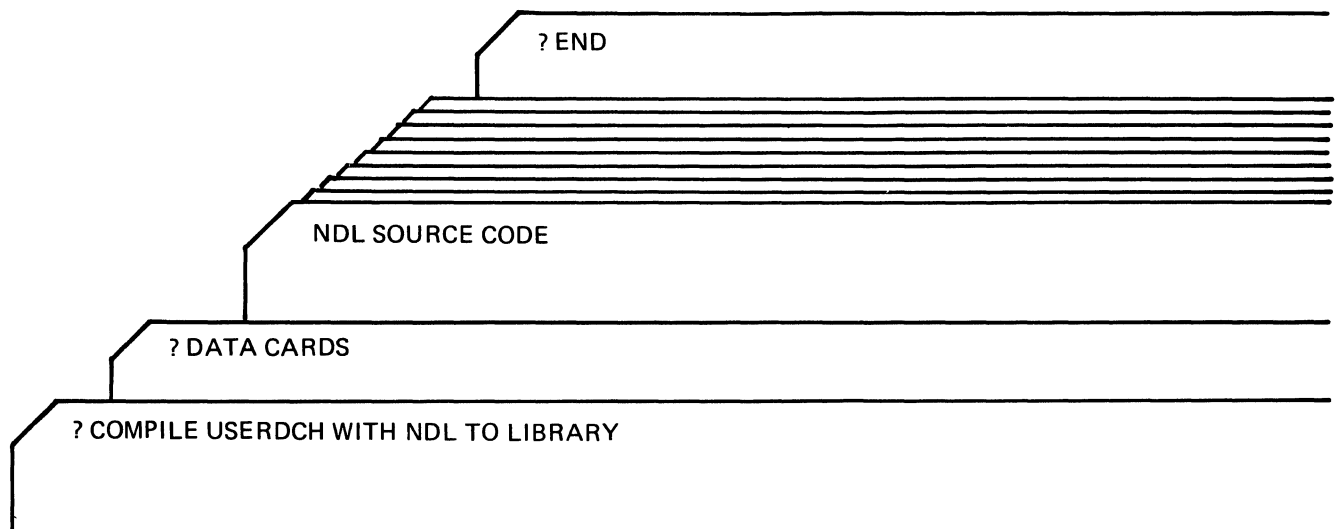


Figure 5-1. NDLS Source Card Deck

Column 72 of a card may be considered logically adjacent to column 1 of the next card. This allows identifiers, numbers, strings, etc. to be continued on the next card.

Comments

A percent sign (%) in any column from 1-72 inclusive results in the compiler skipping to column 1 of the next card. The programmer may enter any desired remarks or comment following the percent sign.

Example:

```
FETCH CHARACTER.    % TAKE 1 BYTE FROM INPUT MESSAGE
```

Percent signs are considered to be included in the NDLS source code if they are within an (EBCDIC STRING), as in this example.

Example:

```
TRANSMIT "LEGAL%".
```

If the "%" is in column 1 the remaining 71 columns of the free-form field are treated as a comment.

Example:

```
% THIS ENTIRE CARD CONTAINS A COMMENT
```

Sequence Numbers

Card columns 73-80 inclusive may be used for sequence numbers. The use of sequence numbers is optional.

COMPILER CONTROL CARDS

Various options are available during compilation and may be activated by \$ control cards. The options cover the areas of list format, error and warning handling, source maintenance, changing stack sizes, and merging source code.

\$ control cards must have the \$ symbol in column 1. The options to be included are to follow the \$, with one or more spaces separating each option specified.

All options are normally “off” except LIST, CHECK, and DOUBLE.

The available options and an explanation of their functions appear alphabetically as follows:

<u>Option</u>	<u>Description</u>
CHECK	This option causes the compiler to print warning messages for sequence errors in the source-language input. A sequence error will occur when the sequence number of the previous card image is greater than or equal to the current sequence number.
CODE	The generated SDL object code will be listed on the line printer when this option is used.
CONTROL	All \$ control cards will be listed on the source program listing.
CREATE LIBRARY	When this card is used, NDL should have been called by an EX NDL rather than by a COMPILE. The REQUESTS and CONTROLS of the new library must follow this card. Execution of the CREATE is terminated by the occurrence of an ?END card. The default name of the new library is NDL/NEWLIBRARY.
DOUBLE	Unless reset, the printer listing of the source program will be double spaced.
DYNAMICSIZE < integer >	This option is used to set the Network Controller’s dynamic memory size to < integer > bits.
FORGETERRORS	This option directs the compiler to generate the object Network Controller despite syntax errors.
LIBINFO	This option lists information about the NDL/LIBRARY being used.
LIBRARY < identifier >	The NDL source code specified by < identifier > is retrieved from the NDL/LIBRARY and inserted in the user’s program following the \$LIBRARY card. The LIBRARY option may not be included on a card containing other options. When the LIBRARY option is used to access standard REQUEST and CONTROL routines the standard REQUESTS must precede the standard CONTROLS.
LIST	The source code will be listed.
LST	The source code will be listed.
MERGE	This option is used to merge the primary input with the secondary input.
NEW	A new source file will be created for use later as secondary input when this option is specified.
NIF	This option causes the NDL Compiler to generate a new Network Information File but not a new object file. Changes in the TERMINAL, STATION, LINE, or FILE Sections are accommodated, but the original REQUEST and CONTROL coding must be the same as in the original compile. The result is a saving in compilation time.
NSSIZE < integer >	The Network Controller’s Name Stack size may be set to < integer > entries with this option.
PAGE	Causes the compiler to skip to the top of a new page when printing.

<u>Option</u>	<u>Description</u>
SEQ	<p>The source may be sequenced by supplying a beginning sequence number and an increment. The numbering will begin at SEQ BASE and will be incremented by SEQ INCRMT. In the following example, note that a plus sign (+) separates SEQ BASE and SEQ INCRMT which are both \langleintegers\rangle .</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> $\\$ \text{ SEQ } \quad \text{ SEQ BASE } + \text{ SEQ INCRMT}$ </div> <p>If only \$ SEQ is specified (SEQ BASE and SEQ INCRMT omitted) the numbering will start with 00000000 and each succeeding number will be incremented by 100. Sequence numbers for \$ control cards are printed when \$ SEQ is specified.</p>
SGL	The listing of the source program on the line printer will be single spaced when this option is chosen.
SINGLE	This option is identical to SGL.
SUPPRESS	Syntax warnings will not be printed on the source program listing when the SUPPRESS option is used.
UPDATE LIBRARY	<p>\langle new library name \rangle</p> <p>When this option is used, NDL should have been called by an EX NDL rather than by a COMPILE. Following this card must be the request and control patches with appropriate sequence numbers. The update is terminated by the occurrence of an ?END card. The default name of the new library is NDL/NEWLIBRARY.</p>
VSSIZE \langle integer \rangle	The Network Controller's Value Stack Size may be set to \langle integer \rangle bits with this option.
VOID	<p>This option may be used in conjunction with \$ MERGE to eliminate certain unwanted secondary source records from the new source file being created.</p> <p>By specifying \$ VOID, the secondary source record with the current sequence number is skipped by the compiler.</p>
VOID \langle integer \rangle	\$ VOID may also be followed by an 8-character integer, which instructs the compiler to skip all secondary source records beginning at the current sequence number and continuing until a secondary source record is read that has a sequence number higher than the 8-character integer specified.

Options may be turned off by specifying \$ NO followed by the name of the option to be turned off. This allows options to be turned on and off at the user's discretion. NO does not affect the VOID or LIBRARY options.

The various options may be grouped on one or more cards with the exception of the LIBRARY option, which must be on one card by itself.

NDL CODING AIDS

There are several coding aids provided by the NDL Compiler that simplify NDL source code preparation for the user.

Compiler-Defined Identifiers

Table 3-2 in the DECLARATION Section lists the compiler-defined identifiers for the more common Data Comm control symbols. For example, the symbols ETX and STX are recognized by the NDL Compiler to equal the hexadecimal strings "03" and "02" respectively.

Default Definitions

The user may reduce repetitious coding by using the default definitions that are available in the **TERMINAL**, **STATION**, **LINE**, and **FILE** Sections.

Library REQUEST/CONTROL Routines

Many users should find it unnecessary to write their own **REQUEST** and **CONTROL** routines, since the **NDL** Compiler maintains a set of library **REQUESTS** and **CONTROLS** to handle common line disciplines and line driving techniques. These routines are included in the **NDL** source language library (**\$ LIBRARY**) and are written in **NDL** source language. At compile time they are read from an **NDL** release disk file and integrated directly into the user's source program. The procedures can be modified if desired by creating a new **NDL** source file. Refer to appendix E.

When the **LIBRARY** option is used to access library **REQUEST** and **CONTROL** routines, the **REQUESTS** must precede the **CONTROLS**.

The following example shows how the library routines could be used in an **NDL** program.

```
?COMPILE NETCON WITH NDL TO LIBRARY
?DATA CARDS
DECLARATION: CONSTANT.
$LIBRARY POLLTCTD           %   Library REQUEST
$LIBRARY FASTSELCTD        %   Library REQUEST
$LIBRARY AUTOPOLCTL        %   Library CONTROL
TERMINAL TC500:
    BUFFERSIZE             = 256.
    ADDRESS                 = 2.
    TRANSMISSION           = 3.
    TYPE                    = 1.           %   TC500
    REQUEST=POLLTCTD:RECEIVE,FASTSELCTD:TRANSMIT.
STATION TC1:                %   TC1 is LOGICAL STATION 1
    TERMINAL                = TC500.
    MYUSE                   = INPUT,OUTPUT.
    RETRY                   = 5.
    CONTROLLER              = FALSE.
    ADDRESS                 = "1A".
LINE TC5LINE:
    CONTROL                  = AUTOPOLCTL
    STATION                  = TC1.
    ADDRESS                  = 7:1:0.
    AUTOPOLL                 = 5.
FILE RMTFILE:
    FAMILY = TC1.           %   TC1 is RELATIVE STATION 1
FINI
?END
```

SECTION 6

NETWORK CONTROLLER DIAGNOSTIC AIDS

GENERAL

Network Controller Diagnostic Aids are provided for problems that occur during the initial start-up phase of a Data Comm system or at a later time. Users of the B 1800/B 1700 Network Definition Language have the following facilities available to aid them in defining and solving problems in the Data Comm system:

- a. Diagnostic REQUEST sets.
- b. Audit facilities.
- c. IOLOG.

DIAGNOSTIC REQUEST SETS

A TERMINAL DIAGNOSTIC REQUEST STATEMENT that specifies an alternate REQUEST for diagnostic purposes can be included in the TERMINAL Section of the NDL source program. The REQUEST specified is activated when a CHANGE REQUEST is executed in a Message Control System.

This diagnostic procedure can be used in an on-line environment. For example, if a station or group of stations is error prone, a diagnostic REQUEST can be invoked to run an on-line diagnostic routine without affecting the remainder of the Data Comm system. The results can be transmitted from the Network Controller to the MCS, or to any peripheral device capable of output.

AUDIT FACILITIES

When real-time debugging of an on-line Data Comm system is desired, the AUDIT statement can be included in one or more REQUESTS. The REQUEST can be modified (not dynamically) to send information about such items as buffers, work area, and message text to a peripheral device. Refer to the DECLARATION Section for additional information concerning the AUDIT statement.

IOLOG

General

The IOLOG feature is included in every NDL-compiled Network Controller. IOLOG is used as a debugging aid for the initial installation or subsequent modification of a NDL Data Comm system. IOLOG monitors all Data Comm line activity (polls, selects, data transfer) and writes pertinent information on a diskfile labelled DC/AUDIT.FILE.

A utility program, DC/AUDIT, is used to read and print the contents of the DC/AUDIT.FILE in a form that is easily understandable. The DC/AUDIT program can be executed concurrently with, or subsequent to, Network Controller execution.

Operating Instructions

To initiate IOLOG the DC/AUDIT program should be running, and a message must be entered at the console printer in the following format:

```
< mix number > AXIOLOG ON  
< > AX END
```

The mix number is that of the Network Controller. A response message is then printed on the console printer indicating that IOLOG was initiated, the date, and the time of day.

OFF

IOLOG can be stopped by again entering < mix number > AXIOLOG at the console printer. A response message is then printed on the console printer indicating that IOLOG was stopped, the date, and the time of day. Data Comm processing continues.

IOLOG can be reinitiated and stopped by subsequent entries of < mix number > AXIOLOG.

SECTION 7

NETWORK CONTROLLER ERROR HANDLING

GENERAL

To assure proper functioning of the Data Comm system, the Network Controller does not continue execution after an irrecoverable error occurs (as distinguished from a line error which can occur during normal operation).

When a run-time error occurs, the following message is displayed on the console printer:

```
RUN ERROR < error number > AT < sequence number >
```

The error number corresponds to one of those in the list of Network Controller run errors that follows. The sequence number corresponds to a sequence number on the Network Controller printer listing. The Network Controller requires either a DS or DP (discontinue processing) console message at this time. If the DP message is entered, the resulting dumpfile can be printed by the NDL/DUMP program. The Data Comm network is brought to a quick, orderly shutdown following the entry of either a DS or DP console message.

NETWORK CONTROLLER RUN ERRORS

The error number, description, and origin of each of the run-time errors possible during execution of a Network Controller follows:

<u>Error Number</u>	<u>Description</u>	<u>Origin of Error</u>
0-99	Not used.	---
100	Parity error.	Memory
104	Skeletal Network Controller incompatible with the MCP.	Controller
109	No poll list.	NDL Code
110	No AUTOPOLL buffer, or message length is zero.	NDL Code
124	Invalid Network Information File.	NDL Compiler
125	Invalid Network Information File.	NDL Compiler
126	Invalid Network Information File.	NDL Compiler
127	Invalid Network Information File.	NDL Compiler
128	Invalid Network Information File.	NDL Compiler
129	The network configuration requires more than 40K bytes of memory for tables.	Controller
130	Main control loop exited in Network Controller.	Controller

<u>Number</u>	<u>Description</u>	<u>Origin of Error</u>
131	Invalid address file.	Controller
132	Table size is too small.	NDL Compiler
133	Table size is too small.	NDL Compiler
135	Station is invalid.	NDL Code
149	Station is attached to another line.	Controller
150	Tried to receive input after shutdown started.	NDL Code
180	REQUEST definition is missing or wrong.	NDL Code
191	CONTROL changed state to a different REQUEST.	NDL Code
193	Line has a station not in station list.	Controller
200	Unexpected remote file.	Controller

NDL/DUMP PROGRAM

General

The NDL/DUMP program is a utility program used for analyzing and printing dumpfiles of NDL-compiled Network Controllers. NDL/DUMP can be used to analyze and print dumpfiles of (a) Network Controllers that experienced a terminating type of error during execution, or (b) are snapshots of currently executing Network Controllers.

Data Formats

The following data formats are used to identify various items when a dumpfile of a Network Controller is printed on the line printer:

- a. Decimal.
- b. Hexadecimal.
- c. Character strings.
- d. Hexadecimal strings.

All data items which are 24 bits or less in length and not absolute addresses are printed as decimal numbers. Fixed fields are also printed as decimal numbers, and are preceded by a plus or a minus sign.

Data items that are surrounded by @ on the printer listing are hexadecimal.

Character fields are printed as character strings.

If character fields contain illegal characters the fields are printed as hexadecimal strings (EBCDIC).

The contents of Data Comm buffers are printed in character format where each printable character is printed as itself. Non-printable characters, such as SOH,STX,ETX, are printed in hexadecimal in vertical format.

Example:

The character string @01C1F102C1C2C303@ is printed as:

```
0A10ABCO
1 2 3
```

Operating Instructions

The NDL/DUMP program analyzes a dumpfile that is on disk when the operator enters the following on the console printer:

```
PM < dumpfile number > NC
```

When the NDL/DUMP program reaches EOJ, the dumpfile is not removed from the Disk Directory. The output of the NDL/DUMP program goes to the line printer.

Dump Analysis

The output printer listing of NDL/DUMP contains information and data concerning the following items, subject to the conditions listed in the description of each item.

- a. The header page contains information similar to that of a non-NDL dump.
- b. Global data items are listed in the order of their declaration in the Skeletal Network Controller.
- c. The TERMINAL tables are printed as defined in the TERMINAL Section of the NDL source program.
- d. The STATION tables are printed as defined in the STATION Section of the NDL source program.
- e. The LINE tables are printed as defined in the LINE Section of the NDL source program.
- f. Queue information appears in the following order:
 1. General input queue.
 2. General output queue.
 3. MCS queue.
 4. Control information queue.

The format of these queues is as follows:

```
@Q.HEAD@ Q.FAMILY.NAME/Q.OFFSPRING.NAME @Q.TAIL@ Q.USER.COUNT
```

- g. Any active REQUESTS or CONTROLS and their segment numbers appear as defined at the end of the NDL output printer listing.
- h. Any local variable items that appear in procedures of the Skeletal Network Controller.

Cross Reference of Variables

Table 7-1 contains names of NDL system status variables and their corresponding names in the Skeletal Network Controller and in the NDL/DUMP output.

Table 7-1. Cross Reference of Variables

NDL System Status Variables	Location in NDL/DUMP Printout	Skeletal Network Controller Variables	Note Number
CHARACTER	LINE.TABLE	LINE.CHAR.BUFFER	
FREQUENCY(INPUT)	STATION.TABLE	INPUT.PRIORITY	
FREQUENCY(OUTPUT)	STATION.TABLE	OUTPUT.PRIORITY	
IODESC	LINE.TABLE	IO.DESCRIPTOR	1
OUTPUTATTACHED	LINE.TABLE	OUTPUT.BUFFER.VALID	
LENGTH(INPUT)	LINE.TABLE	INPUT.BUFFER.LENGTH	
LENGTH(OUTPUT)	LINE.TABLE	DCW.SIZE	
LINE	LINE.TABLE	NUMBER	
LINE(CONTROL KEY)	LINE.TABLE	CONTROL.KEY	
LINE(QUEUED)	LINE.TABLE	QUEUED	2
MAXSTATIONS	LINE.TABLE	MAX.STATIONS	
RESULTDESC	LINE.TABLE	IO.RESULT	3
RETRY	STATION.TABLE	RETRY.COUNT	
STATION	LINE.TABLE	LINE.CUR.STATION.NO	
STATION(ENABLED)	STATION.TABLE	ENABLED	
STATION(MYUSE)	STATION.TABLE	MYUSE	
STATION(QUEUED)	GLOBAL ARRAY	STATION.Q	4
STATION(READY)	STATION.TABLE	READY	
STATION(TYPE)	STATION.TABLE	CONTROLLER.LINK	5
STATION(VALID)	STATION.TABLE	VALID	
TERMINALTYPE	TERMINAL.TABLE	TYPE	
TRAN(RECEIVE)	STATION.TABLE	INPUT.TRAN.NO	

Table 7-1. Cross Reference of Variables (Cont)

NDL System Status Variables	Location in NDL/DUMP Printout	Skeletal Network Controller Variables	Note Number
TRAN(TRANSMIT)	STATION.TABLE	OUTPUT.TRAN.NO	6
ERROR FLAGS	GLOBAL VARIABLE	RESULT.DESRIPTOR.ERRORS	
TOG INDEX	STATION.TABLE	STATION TOGGLES	
LINE(TOG[< INDEX >])	LINE.TABLE	LINE TOGGLES	
TALLY[< INDEX >]	STATION.TABLE	STATION TALLYS	
LINE(TALLY[< INDEX >])	LINE.TABLE	LINE TALLYS	
TIME(TALLY)	STATION.TABLE	TIME.TALLY	

Explanation of Note Numbers: The note numbers in Table 7-1 refer to the following explanations:

1. A part of each of the first two descriptors is stored in this field. Each of these parts consists of the following consecutive 24-bit fields:

IO.ACTUAL.END

IO.RESULT

IO.LINK

IO.OP

IO.BEGIN

IO.END

2. If the value of this field is greater than zero, the QUEUED is true; otherwise, QUEUED is false.
3. This field is a concatenation of the IO.RESULT fields of the first two IO.DESRIPTOR fields.
4. If the value of this field is greater than zero, STATION(QUEUED) is true; otherwise, STATION(QUEUED) is false.
5. The following are the possible values and corresponding meanings of this field:

@O@ means NORMAL

@F@ means CONTROLLER

anything else means SLAVE

6. The specific error flags are as follows:

<u>Error Flag Bit Number</u>	<u>Error Type</u>
1	PARITY
2	ACCESSERR
3	Not used
4	TIMEOUT
5	READ
6	ENDOFBUFFER
7	LOSSOFDSR
8	LOSSOFCARRIER
9	ADDERR
10	TRANERR
11	FORMATERR
12	Not used
13	EXCEPTION

Table 7-2 shoes the correlation between the NDL-defined attributes and the field names in the NDL/DUMP program.

Table 7-2. Correlation of NDL Attributes and NDL/DUMP Field Names

NDL Source Statement	Field Name in NDL/DUMP
TERMINAL < TERMINAL IDENTIFIER >	TERMINAL Table Identifier
BUFFERSIZE	BUFFER.SIZE
DIAGNOSTIC REQUEST	DIAG.OUTPUT.REQUEST(DIAG.INPUT.REQUEST)
REQUEST	OUTPUT.REQUEST(INPUT.REQUEST)
TYPE	TYPE
TRANSMISSION NUMBER	TRAN.NO.SIZE
ADDRESS	OUTPUT.ADDR.SIZE(INPUT.ADDR.SIZE)
STATION < STATION IDENTIFIER >	STATION Table Identifier
CONTROLLER	CONTROLLER
FREQUENCY	INPUT.PRIORITY(OUTPUT.PRIORITY)

**Table 7-2. Correlation of NDL Attributes and NDL/DUMP
Field Names (Cont)**

NDL Source Statement	Field Name in NDL/DUMP
LOGICALACK	LOGICAL.ACK
MYUSE	MYUSE
ADDRESS	INPUT.ADDRESS(OUTPUT.ADDRESS)
TERMINAL	TERMINAL.LINK
RETRY	RETRY.COUNT
LINE < LINE IDENTIFIER >	LINE Table Identifier
AUTOPOLL SIZE	POLL.LIST.LENGTH
CONTROL	CONTROL.LINK
STATION	STATION.LINK
TYPE	TYPE

The arrays declared in the global declaration section of the Network Controller contain information regarding which stations belong to which family of stations. The information can be found in the NDL/DUMP output by using the following method:

- a. Find the family name in the FILE.NAME array.
- b. Find the index into the FILE.NAME array that points to the given family name. The index is in parentheses next to the file name, and is the family number.
- c. The station number is the index into the station identifier list in the NDL FAMILY statement. Note that this index starts with a value of zero.
- d. FILE.STATION.LIST.PTR (family number) is the station pointer.
- e. FILE.STATION.LIST (station pointer + station number) is the station list pointer.
- f. STATION.LIST (station list point - 1) is the index into the STATION table.

APPENDIX A

RESERVED WORDS

The following list contains all of the reserved words known to the B 1700 NDL Compiler. They may not be used as identifiers.

ACCESSERR	FALSE	PAGE
ACU	FAMILY	PARITY
ADAPTER	FETCH	PASSIVE
ADERR	FILE	PHONE
ADDRESS	FINI	POLL
AND	FINISH	QUEUED
ASC63	FOR	READY
ASC64	FORMAT	RECEIVE
ASC68	FORMATERR	REMOTE
AUDIT	FREQUENCY	REQUEST
AUDITFILE	GE	RESIDENT
AUTOANSWER	GT	RESULTDESC
AUTOPOLL	HEADER	RETURN
BREAK	HORIZONTAL	RETRY
BUFFER	IDLE	SCREEN
BUFFERSIZE	IF	SEC
BUFOVFL	INCREMENT	SECURITY
CALL	INITIALIZE	SEQUENCE
CANCEL	INITIATE	SIGNAL
CARRIAGE	INPUT	SPECIAL
CASE	INPUTATTACHED	SPO
CHAR	IODESC	STATION
CHARACTER	LE	STORE
CLEAR	LENGTH	SUM
CODE	LINE	TALLY
COMMENT	LOGICALACK	TASK
CONSTANT	LOGIN	TERMINAL
CONTINUE	LOSSOFCARRIER	TERMINALTYPE
CONTROL	LOSSOFDSR	TERMINATE
CONTROLLER	LS	TEXT
CONVERT	LT	THEN
DEBUG	MAX	TIME
DECIMAL	MAXSTATIONS	TIMEOUT
DECLARATION	MEMORY	TO
DECREMENT	MIN	TOG
DEFAULT	MODE	TRACE
DEFINE	MYUSE	TRAN
DIAGNOSTIC	NE	TRANERR
DIALIN	NIF	TRANSMISSION
DIALOUT	NO	TRANSMIT
DISCONNECT	NORMAL	TRUE
DISPLAY	NOT	TURNAROUND
DO	NULL	TYPE
DUMMY	NUMBER	UNDO
DUMP	ODD	USER
EBCDIC	OR	VERTICAL
ELSE	OUTPUT	
ENABLEINPUT	OUTPUTATTACHED	
END		
ENDOFBUFFER		
EQ		
ERROR		
EVEN		
EXCEPTION		

APPENDIX B

GLOSSARY

ACK

A control character that is used as an affirmative response to a normal selection (indicating “ready-to-receive”) or a transmission (indicating “message accepted”). As an affirmative response to a selection, ACK may optionally be preceded by station identification, AD1, AD2, or other information such as a reply.

ACK0, ACK1

These replies, when in proper sequence, indicate the the previous block received was accepted without error, and that the receiver is ready to accept the next block of the transmission. ACK0 is the positive response to selection (multipoint), or line bid (point-to-point). The alternate use of ACL0 and ACK1 is used for affirmative replies. The use of ACK0 and ACK1 provides a sequential checking control for a series of replies. Thus, it is possible to maintain a continuous check to ensure that each reply corresponds to the immediately preceding message block. The affirmative response to a poll is the transmission of a message.

ACK0 is represented by a DLE character followed by a hexadecimal 70.
ACK1 is represented by a DLE character followed by a hexadecimal 61.

AD1, AD2

A two-character address, established as the address of a device at a terminal. These two characters are used to address a terminal in polling or selection, and used in the message header to identify the terminal from which a message is transmitted. This address may also be used as an identification prefix to acknowledge (ACK) that a terminal is ready to receive a message or to identify NAK. On receipt of a message, the receiving station may use AD1-AD2 to verify that the message originated at the polled terminal. For group addressing of broadcast to all terminals, AD1-AD2 indicates the terminal that will acknowledge receipt of the message. In systems which preclude “downstream” (terminal-to-terminal) communication, AD1-AD2 in the header, transmitted by the central computer, may be defined to represent the terminal address and is used for address checking.

ASCII (American Standard Code for Information Interchange)

This code, established as an American standard by the American Standards Association, defines codes for a set of characters to be employed in the interchange of information between business equipment over telephone and telegraph circuits. The code consists of 128 control and graphic characters.

Attach Initiator

The file that writes an ATTACH.

Application Program

A program that processes data and is usually unique to one type of application.

Automatic Calling Unit (ACU)

A device that may be furnished by a communications carrier allowing a business machine to automatically establish a dialed link over the communications network.

Autopoll

A technique used by the central computer to interrogate terminals to determine if they have anything to transmit without interrupting the Network Controller for the initiation of each station poll.

Asynchronous transmission

A means of transmission which requires that there always be an integral number of unit intervals within a character between any two significant instants in time. However, this need not be so between two significant instants in time in an integral number of unit intervals in different characters.

Bandwidth

An expression of Hertz measurement which states the difference between the high and low frequencies in a communications channel.

Baud

A unit of signaling speed equal to the number of discrete conditions or signal events per second.

Baudot code

A code used in the transmission of data in which five bits represent one character. Named for Emile Baudot, a pioneer in printing telegraphy, it is sometimes referred as five-bit code, five-channel code, five-unit code, "Teletype" code, although a "Teletype" unit is no longer limited to its use.

Binary Synchronous (BSC)

A method of data transmission which allows sending or receiving of data streams. The data streams may contain bit patterns which would normally be detected as control character sequences. Therefore, data streams containing object program code may be transmitted. The transmission is always synchronous.

Bit rate

The speed, usually expressed in bits per second (BPS), with which bits are transmitted over a communication channel.

Bit synchronization

The process by which the transmitting and receiving bit frequencies are made substantially the same and are maintained, by means of correction if necessary, in a desired phase relationship.

Block

A group of characters or bits sent as an integral unit. Usually an error-checking procedure is applied over a block, for control and recovery purposes.

Block Check Character (BCC)

The Block Check Character is a redundant character added to the end of a transmission block for the purpose of error detection.

Block number (BL#)

An option which may be used when data must be subdivided into separate units for transmission. The block number consists of a two-character number identifying the sequential block number in a blocked message. The first character is always a DLE character.

Break

The facility to enable a receiving device to interrupt the transmitting device and, by doing so, to be able to take control of the circuit.

Broadcast

The simultaneous sending of a message to several stations of a network.

Broadcast Select (BSL)

The Broadcast Select control code is used to indicate "This is a broadcast message" to all stations. In the broadcast sequence. AD1-AD2 identify the station which will acknowledge receipt of the message. Broadcast Select is followed immediately by a transmission block, without requiring acknowledgement of the selection.

Burroughs Data Link Control (BDLC)

BDLC is a Data Comm link-level control procedure adaptable to a variety of system implementations. BDLC can produce configurations compatible with other link-level protocols available.

Carrier system

A means of obtaining a number of channels over a single path by modulating each channel upon a different carrier frequency and demodulating at the receiving point to restore the signals to their original form.

Cathode Ray Tube (CRT)

A large electron tube used to emit electrons onto a phosphor screen, thus creating a visual display.

Central office

The place where a common carrier has equipment which interconnects customer transmission lines.

Centralized operation

The control discipline used in a multipoint data communication network in which all message transfers must involve and control station. Transmissions directly between tributary stations are not allowed.

Channel

The communication path used to transmit signals between two or more points. Often referred to as a link, circuit, line, or path.

Character

A set of elements arranged in orderly groups to represent digits, symbols, or letters. Represented in two forms: (1) for use by computers, business machines, communications facilities, etc., usually in groups of binary bits; and (2) for use by man in conveying an understandable form of decimal digits, alphabetic characters, punctuation, or other special symbols. Characters may be represented using groups of bits, commonly five, six, seven, or eight bits.

Character synchronization

A process in which the character frequencies of the transmitting and receiving ends of a transmission circuit are maintained in a phase relationship in order that the receiver can derive the transmitted characters from the signals received.

Circuit

The configuration of equipment used in transmitting data from one location to another. A circuit may involve more than one type of facility.

Circuit assurance

The function of verifying the existence or the operational state of the communication channel between stations.

Code

A system of symbols and rules for use in representing information.

Common carrier

A company that provides communication service for public hire.

Communication link

The connection of two or more stations by the same communication channel. This link includes the communication control capability of the stations connected in the link.

Communication system

The combination of all the links, link interface equipment, application and systems software, including control procedures, that are required to effect the transmission of coded information between stations in the system.

Conditioning

Private and leased transmission lines can be conditioned to reduce distortion and thereby provide data transmission at lower error rates.

Connection

The established path between two or more terminal installations. A permanent connection is established by using switching facilities.

CCITT (Consultative Committee International Telegraphic and Telephonic)

An international committee that defines telegraphic and telephonic interconnections and switching standards.

Contention (CON)

The contention character instructs all terminals receiving this character to go to the contention mode. The contention mode is an operational condition on a data communication link in which no station is designated as a master station. With contention mode, each station on the link must monitor the signals on the link and wait for a quiescent condition before initiating a bid for master status.

Continuous operation

A type of message transmission in which the master station need not stop for a reply after transmitting each acknowledgement unit, but may continue transmission with the next acknowledgement unit.

Control character, data communications

A functional character intended to control or facilitate transmission of information over communication networks. The major communication control characters provided are ACK, BEL, CAN, CR, DC1, DC2, DC3, DC4, DEL, DLE, ENQ, EOT, ESC, ETZ, FIB, FF, FS, FSL, GS, GSL, HT, LF, NAK, NUL, POL, RS, SI, SO, SHO, STX, SYN, US, and VT.

Control procedure, data communications

The means used to provide for the orderly communication of information between the stations on a data communications link.

Control state

An operational state on a data communications link in which characters, other than communication controls, may be given control interpretations. A data link is in the control state when message transfer is not in progress.

Control station

A permanently designated (unaffected by link control procedures) station on a data link with the overall responsibility for the orderly operation of the link. A control station generally has additional control capabilities (e.g., the capability to poll other stations) not provided at other stations, and is designated to initiate error recovery procedures in the event of certain abnormal conditions on the link.

Conversational mode

An operational mode in which message information is used in lieu of, or in addition to, control characters as replies for message information.

(CRF) Controlling Remote File

The file with HEADERS to which a station was most recently assigned, either with an ATTACH or with an OPEN. If the primary file has HEADERS, it is the CRF; otherwise, the secondary file is the CRF.

Cyclic Redundancy Check (CRC-16)

A 16-bit redundant character added to the end of a transmission block for the purpose of error detection and control. All characters following STX (or SOH) except SYN and the first DLE of a DLE DLE sequence are included in the CRC accumulation. A cyclic redundancy check is a division, performed by both the transmitting and receiving stations, using the numeric binary value of the message as a dividend which is divided by a fixed polynomial. The most common polynomial used is $X^{16}+X^{15}+X^n+1$. The quotient is discarded, and the remainder serves as the check character, which is then transmitted as the Block Check Character (BCC) immediately following a check-point character (ITB, ETB, or ETX). The receiving station compares the transmitted remainder to its own computed remainder, and finds no error if they are equal.

Data communications system

One or more data links, each of which may be operating in the same or a different mode.

Data communications

The transfer of encoded information by means of electrical transmission systems.

Data link

This is an ensemble of terminal installations and the interconnect network, together with controlling procedures operating in a particular mode, that permits information to be exchanged between locations.

Data Link Escape (DLE)

Data Link Escape is a control character that is used to change the meaning of a limited number of contiguously following characters, which then become supplementary data transmission control functions. Only graphics and transmission control characters can be used in DLE sequences.

The Data Link Escape character combinations are used primarily for binary synchronous operation. The Data Link Escape character is always the first character of the two character combinations. The DLE combination characters are listed below:

ACK0	}	These characters are used only for binary synchronous operation. Refer to this Glossary for a detailed description of each of these characters.
ACK1		
RVI		
WACK		
DLE STX		Used to start transparent mode in BSC operation.
DLE DLE	}	Used as control characters during transparent mode of operation. The mnemonic for DLE EOT is DEOT.
DLE EOT		
DLE ENQ		
DLE SYN		
DLE ETB	}	These characters are used to end transparent mode of operation.
DLE ETX		
DLE ITB		

Data set

A circuit termination device used to provide an interface between a data communication circuit and a data terminal. A modulation and/or demodulation function is typically performed in a data set.

Data terminal

That part of a station concerned with the functions of generating data and/or recording or displaying of data, together with the control equipment or system software necessary to control these functions.

Dedicated channel

A communication channel provided for the exclusive use of a specific subscriber on a contract basis. (See Private line.)

Delimiters

Control signals used to define the extent of a particular sequence of characters.

Demodulation

The process of retrieving an original signal from a modulated carrier wave. This technique is used in data sets to make communication signals compatible with business machine signals (contrast with definition of Modulation.)

DEOT

The mnemonic for the DLE EOT communication control sequence, used to signal that a disconnect of a switched circuit must be initiated.

Dial-up

The use of a dial or pushbutton telephone to establish a station-to-station telephone contact.

Disconnect (DEOT)

A Disconnect is a communication control sequence consisting of DLE followed by EOT, used to signal that a disconnect of a switched circuit must be initiated.

Display unit

A device which provides a visual representation of data (see Cathode ray tube).

Duplex

In communications, pertains to the capability of simultaneous two-way transmission.

End of Text (ETX)

The End-of-Text control character is used to indicate the end of the last block of text.

End of Transmission (EOT)

The End-of-Transmission control character is used to indicate the conclusion of a communication sequence. Receipt of an EOT will set the terminal in a control state, waiting for a polling, selection, or contention sequence. EOT may be transmitted by a master station to abort a transmission sequence. To insure that terminals are in a control state, EOT precedes a communication control sequence. EOT is transmitted by a remote terminal as a “no traffic” response to a poll.

End of Transmission Block (ETB)

The End-of-Transmission Block control character is optionally used when messages are of sufficient length to warrant their being broken into smaller transmission blocks. ETB indicates the end of a block of data, either in a heading or in the message text. The heading or text is resumed after transmission of a block number and SOH or STX.

Enquiry (ENQ)

The Enquiry control character is used as a request for a reponse of station status or for a retransmission of control characters. This character is also used as the final character of a poll or in a selection when response from the other station is required.

Error

Any received character or sequence of characters that does not conform to those transmitted.

Error control

A system that detects the present of errors. In some systems, refinements are added that correct some of the errors, either by operations on the received data or by retransmission of the data from the source.

Error recovery procedure

Data communications control procedures used to restore normal operation to a data link after unusual (abnormal) events have occurred.

Exchange

A defined area, served by a communications carrier, within which the carrier furnishes service.

Extended Binary Coded Decimal Interchange Code (EBCDIC)

This code is a transmission code which can represent up to 256 different characters.

Facility

The type of communication medium used to provide communication circuits (e.g., cable, radio, open-wire).

Facsimile (FAX)

Transmission of pictures, maps, diagrams, etc. The image is scanned at the transmitter, reconstructed at the receiving station, and duplicated on some form of paper.

Family

A multistation file.

Family population

The number of active stations in the family of the file. Family size is the number of stations assigned to the file.

Fast Select (FSL)

The Fast Select character is used to indicate “This is a fast select,” in a selection sequence transmitted by the central computer. Fast Select is followed immediately by a transmission block, without requiring acknowledgement of the selection.

Filler

A character used as a time-fill or space-fill when a block of a specified size is required and the heading and/or text characters are of insufficient length for the requirement.

Format

The predetermined arrangement of characters, field, lines, page numbers, punctuation marks, etc., used to transfer data from one location to another. Refers to input, output and files.

Full-duplex channel

A communication channel where the signaling may be in both directions simultaneously. The signaling speeds used for the two directions of transmission on a full-duplex channel need not be the same.

Full-duplex transmission

A type of transmission where information is sent in both directions simultaneously.

Group Select (GSL)

The Group Select character is used to indicate “This is a message for a group of stations.” In the group select sequence, AD1-AD2 identifies the station which will acknowledge receipt of the message. Group Select is followed immediately by a transmission block, without requiring acknowledgement of the selection.

Half-duplex channel

A communication channel where the signaling may be in either direction, but not in both directions at the same time.

Half-duplex transmission

A type of transmission where information is sent in one direction or the other direction, but not in both directions simultaneously. (Refer to two-way alternate transmission.)

Handshaking

This is the exchange of control signals between data sets when the connection is established.

Header

A sequence of characters that may precede the text of a message to provide the information necessary to route the message to its ultimate destination(s). A message header may also contain communications relating information other than routine instructions. A message header is preceded by an SOH character and is ended by an STX character.

Headers

An option on a remote file which allows system control functions and provides a 50-byte header on all data messages moving through that remote file.

Hertz (Hz)

A measure of frequency or bandwidth. The same as cycles per second.

Holding time

The length of time a communication channel is in use for each transmission. Includes both message time and operating time.

Identification

A sequence of characters used during the establishment of a connection and subsequently, if needed, to identify a station.

Identifier

A sequence of one or more characters transmitted by a station in order to identify itself.

Information block

A sequence of characters of fixed or variable length which is a subdivision of an information message formed for the purpose of meeting transmission requirements.

Information message

A sequence of characters conveying the text. It may also convey supplementary information forming a heading.

In-plant system

A data-handling system confined to one building or a number of buildings in one locality.

Intermediate Transmission Block End (ITB)

The ITB control character is used for binary synchronous operations to divide a message (heading or text) for error-checking purposes, without causing a reversal of transmission direction. The block-check character immediately follows ITB and resets the block-check count. After the first intermediate block, successive intermediate blocks need not be preceded by STX or SOH. (For transparent data, which may contain control character sequences, each successive intermediate block must begin with DLE STX). If one intermediate block is heading and the next is heading and the next intermediate block is text, STX must begin the text block.

Intermediate Transmission Block End (ITB) cont.

Normal line turnaround occurs after the last intermediate block, which is terminated by ETB or ETX (DLE ETB or DLE ETX for transparency). When one of these ending characters is received, the receiving station responds to the entire transmission. If a block-check error is detected for any of the intermediate blocks, a negative reply is sent, which requires retransmission of all intermediate blocks.

All bisynchronous stations must have the ability to receive ITB and its attendant BCC. The ability to transmit the ITB character is a station option.

Line adapter

A line adapter consists of a logic card used to interface a data set or communications line into a Multi-Line or Single-Line I/O Control.

Line switching

A switching technique that temporarily connects two lines together so that two stations can directly exchange information.

Link, data communication

The logical association of two or more stations interconnected by the same communication channel. A data link includes the communication control capability of the connected stations.

Local channel

A channel connecting a communications subscriber to a central office.

Logical Station Number (LSN)

The number by which the Network Controller uniquely identifies a station for normal transactions. LSNs begin with 001 and proceed sequentially through all the stations declared in the STATION section of the NDL source program.

Master station

A station temporarily designated (can be changed by the link control procedures) to have control of the data link at a given instant. Master status is normally conferred upon a station so that it may transmit a message.

Message

A sequence of characters arranged in a form suitable for the purpose of conveying information from an originator to one or more destinations (or addresses). It contains the information to be conveyed (called the text) and may, in addition, contain communication information to aid in the routing of handling of the message (called the header).

Message Control System (MCS)

Any program which opens a remote file with the HEADERS option and thereby controls the stations in that remote file.

Message format

Rules for the placement of such portions of a message as message header, address, text, and end of message.

Message numbering

The identification of each message within a communications system by the assignment of a sequential number.

Message switching

The technique of receiving a message, storing it until the proper outgoing circuit and station are available, and then retransmitting toward its destination.

Microwave

All electromagnetic waves in the radio frequency spectrum above 890 megahertz.

Modem

See definition of Data set.

Modulation

The process by which some characteristic of one wave is varied in accordance with another wave. This technique is used in data sets to make business machine signals compatible with communication facilities (contrast with definition of Demodulation).

Most Significant Bit (MSB)

The most significant bit is the bit of a binary number which is assigned the greatest numeric value. It is generally assigned to the most significant bit position, as in the decimal numbering system.

Multidrop

See Multipoint network.

Multi-Line Control (MLC)

The Multi-Line Control is an I/O control that provides the functional control between the computer system and the line adapters. An integral part of a multi-line data communications subsystem.

Multiple address message

A message to be delivered to more than one destination.

Multiplexing

The division of a transmission facility into two or more channels.

Multipoint circuit

A circuit interconnecting several stations (see definition of Multipoint network).

Multipoint network

(1) A data communication line connecting three or more stations; (2) a data communication link with the control capability necessary to interconnect three or more stations.

NAK

The Negative Acknowledgement control character is used as a negative response to a normal selection (indicating not ready to receive) or a transmission (indicating character parity failure for any character of a block or, in a message, a failure of the BCC, or block number sequence check failure). NAK may optionally be preceded by station identification AD1, AD2, or other information.

Network

The ensemble of equipment through which connections are made between terminal installations. These equipments operate in real time and do not introduce, store, or forward delays. A switched telephone network is the network of telephone lines normally used for dialed telephone calls. A private network is a network of communication channels reserved for the use of one customer.

Network Controller (NC)

The program generated through compilation of a Network Definition Language source program. The Network Controller handles the line discipline for the data communication devices of a system and the interface queue between an MCS and the operating system.

Network Definition Language (NDL)

A descriptive, free-form language for defining and implementing a data communications network. The NDL compiler analyzes the input statements and generates a custom Network Controller.

Network Information File (NIF)

Contains tables that describe the physical and logical attributes of the network. The initialization values for the LINE, STATION, TERMINAL, and FILE tables of the Network Controller are contained in the NIF.

Non-centralized operation

A control discipline for multipoint data communication links in which transmissions may be between tributary stations or between the control station and a tributary station(s).

Off line

Used to describe terminal equipment which is not connected to a transmission line. Can also describe other devices not in direct communication with the central processing unit.

On line

Used to describe terminal equipment which is connected to a transmission line. Can also describe other devices in a direct communication with the central processing unit.

Pad character (PAD)

To insure that the first and last characters of a transmission are properly transmitted by the data set, all binary synchronous stations add a PAD character before and after each transmission. The leading PAD character consists of a hexadecimal "55" character SYN character, the trailing PAD character consists of all "1" bits (hexadecimal "F-F" character). Although the PAD character is comprised of eight bits, the receiver generally only checks the first four bit positions.

Parallel transmission

The simultaneous transmission of a certain number of signal elements constituting the same telegraph or data signal. For example, all bits of a character may be sent simultaneously in parallel transmission. (Contrast with definition of Serial transmission).

Parity bit

A bit associated with a character or block for the purpose of checking for the presence of error within that character or block. This bit is chosen to make the modulo 2 sum of the bits (including the parity bit) in the character or block a zero (even) or a one (odd) as required by the system.

Parity check

A check that tests whether the number of ones (or zeros) in an array of binary digits is odd or even.

Passive station

A station on a data link that is (temporarily) neither a master station nor a slave station.

Point-to-point connection

A configuration in which a connection is established between two, and only two, terminal installations. The connection may include switching facilities.

Poll (POL)

The Poll character is used to indicate "this is a poll," preceding ENQ in a polling sequence.

Polling

A technique for assignment of master status to a particular station on a Data Comm link. Polling is centrally controlled in order to maintain a strict discipline over the operation of a number of points.

Polling supervisory sequence

A supervisory sequence that performs a polling function.

Prefix

A sequence of characters (other than communication controls) used in a supervisory sequence to define or qualify the meaning of the supervisory sequence.

Primary File

The file to which a station was most recently attached or included in an OPEN. Normal input goes to the primary file.

Private line

A channel or circuit furnished a subscriber for his exclusive use (see definition of Dedicated channel).

Queue

A linear list for which all insertions are usually made at one end of the list and all deletions and other accesses are made at the other end.

Recovery procedure

A process by which a responsible station within the network attempts to resolve either conflicting or erroneous conditions arising in the communication process. The control or master station is responsible for this procedure.

Redundancy

The portion of the total information contained in a message which can be eliminated without loss of essential information.

Relative Station Number (RSN)

The number by which a user program with a remote file using a remote key uniquely identifies a station. An RSN equal to 0 implies a control message. An RSN equal to 1 implies the first station in the file's FAMILY statement. RSNs proceed sequentially through the stations delineated by a file's FAMILY statement, except that a Controlling Remote File may modify the LSN-LIST and thereby modify the RSNs of the remote file. If a station is detached from a file with a remote key, the RSNs remain unchanged. If a station is attached to a file with a remote key, the new RSN is the same as the old RSN if the file was attached previously; otherwise, the RSN is one larger than the greatest RSN previously associated with the file.

Remote File

A file declared in a program which, in conjunction with the Network Controller, provides input, output, or I/O with a set of data communication devices.

Remote Job Entry (RJE)

A computer communications system in which a central computer executes programs that were sent to it by other computers that appear as terminals.

Reverse channel

A communication channel between a slave station and a master station; used exclusively for control signals.

Reverse Interrupt (RVI)

The reverse interrupt control character is used only for binary synchronous operation. The RVI control sequence is a positive response used in place of the ACK0 or ACK1 positive acknowledgement. RVI is transmitted by a receiving station to request termination of the current transmission for one of the following reasons: (1) The receiving station must transmit a high-priority message to the sending station; (2) The control station, when in a multipoint environment, acts as a receiver and wants to communicate with another station on the line. Successive Reverse Interrupt control characters cannot be transmitted, except in response to ENQ.

The sending station treats the RVI as a positive acknowledgement and responds by transmitting all data that prevents it from becoming a receiving station. More than one block transmission may be required to empty the buffers of the sending station.

The ability to receive RVI is mandatory for all binary synchronous stations, but the ability to transmit RVI is optional. RVI is represented by a DLE character followed by a hexadecimal 6B.

Recommended Standard 232C (RS232C)

This standard, provided by the Electronic Industries Association, is the recommended interface between data communications terminal equipment and data communications equipment using serial binary data interchange.

Secondary File

The file to which a station was just previously attached or included in an OPEN. The secondary file must have HEADERS, and must have approved the ATTACH or OPEN of the primary file. Input whose first character matches the signal character (not blank) designated in the primary file's ATTACH or OPEN goes to the secondary file.

Select (SEL)

The Select character is used to indicate “this is a normal select,” when it precedes ENQ in a selection sequence.

Selection

A technique for assignment of slave status to a particular station on a Data Comm link.

Selection supervisory sequence

A supervisory sequence that performs a selection function.

Selective calling

The ability of a transmitting station to specify which station or stations on the same line are to receive a message.

Sequential select (SEQ)

The sequential select character is used to indicate that a group of remote terminals is being selected to receive a message addressed to that group. The last terminal selected in the group will acknowledge receipt of sequential select. Sequential select must be followed by AD1-AD2 of another terminal.

Serial transmission

Transmission, at successive intervals, of the signal elements constituting the same telegraph or data signal. The sequential elements may be transmitted with or without interruption provided that they are not transmitted simultaneously. (Contrast with definition of Parallel transmission.)

Shutdown time

The time at which the Network Controller performs several functions necessary to the operation of the Data Comm network. The major functions are the emptying of all output queues, disabling any station input in progress, and issuing termination notification to all executing application programs or Message Control Systems. These functions must be performed before the Network Controller terminates itself.

Signal-conversion equipment

That part of the terminal installation belonging to the data channel, comprising at least one modulator or one demodulator. This equipment provides modulation according to the signals to be transmitted, and/or demodulation of the signals received.

Simplex channel

A communication channel where the signaling may be in one direction only.

Single Line Control (SLC)

The Single Line Control is an I/O control that provides the functional control between a computer system and a line adapter. The SLC is an integral part of a single line data communications subsystem.

Single-address message

A message to be delivered to only one destination.

Slave station

A station that has indicated its readiness to receive a message. The assignment of slave status is temporary and continues for the duration of a transmission.

Start Of Header (SOH)

The Start-Of-Header control character is required only when a message header is to be sent with a transmission. When used, SOH is the first of a sequence of characters used for the message header. The header also may contain terminal identification AD1, AD2, and may, under definition for the specific application, contain other information pertinent to the transmission number (Xm#). A header is ended by STX.

Start of Text (STX)

The Start-Of-Text control character precedes a sequence of characters which form the text of the transmission. STX terminates a header.

Start/Stop transmission

Start/Stop transmission is an asynchronous transmission in which a group of code elements corresponding to a character signal is preceded by a start signal. The start signal prepares the receiving mechanism for the reception and registration of a character and is followed by a stop signal. The stop signal brings the receiving mechanism to rest in preparation for the reception of the next character.

Station

The aggregate of the terminal equipment (and system software), and communication control equipment (and system software) attached to a particular line adapter for and SLC or MLC. Used for the input or output of information for the communications system of which it is a part.

Stop element (in a Start/Stop system)

Binary element that indicates to the terminal installation the completion of receipt of a character to bring the receiving mechanism to rest in preparation for receipt of the next character.

Store-and-forward

Process of message handling used in a message-switching system.

Stunt box

A device to control the non-printing functions of a teleprinter terminal. Control characters can be sent to the stunt box over the communications channel.

Supervisory sequence

A sequence of communication control and, possibly, non-communication control characters that perform a defined control function. A supervisory sequence consists of a prefix together with one or more control character delimiters.

Switching

Operations involved in interconnecting circuits in order to establish a temporary communication channel between two or more stations.

Synchronous idle (SYN)

The synchronous idle control character is used only by a synchronous transmission system in the absence of any other character to provide a signal for establishing and retaining a synchronism. On initiation of synchronous transmission, a number of SYN characters are transmitted prior to transmission of any other character to permit the receiving station to acquire character synchronization.

SYN is also used as a “time fill” when no other characters are available for transmission at any point in a character sequence except between ETB or ETX and the next following BCC. SYN is purged at the receiving terminal and is not included in the summation for BCC.

Synchronous transmission

A transmission process in which there is always an integral number of unit intervals between any two significant instants.

Telegraphy

A system of communication for the transmission of graphic symbols, usually letters or numerals, by use of a signal code.

Telegraphy, printing

A method of telegraph operation in which the received signals are automatically recorded in printed characters.

“Teleprinter”

Term used to refer to the equipment used in a printing telegraph system. A “teletypewriter”.

Transparent mode

An operational mode in which all coded combinations of eight-bit characters are allowed as message characters within a message text.

Tributary station

All stations, other than the control station, which are on a non-switched multipoint network are called tributary stations.

Turnaround time

This is the time required by data sets when switching between transmit and receive modes.

Two-Way Simultaneous Transmission

A type of transmission where information is sent in both directions simultaneously. (See Full-duplex.)

Two-Way Alternate Transmission

A type of transmission where information is sent in one direction or the other direction, but not both directions simultaneously. (See Half-duplex.)

Unattended operation

The automatic features of a station's operation that permit the transmission and reception of messages without a computer operator being in attendance.

USASCII (United States of America Standard Code for Information Interchange)

The USA standard code for information interchange. (See definition of ASCII.)

User Program (UP)

Normally denotes a program which has opened or is opening a remote file without HEADERS.

User Programming Language (UPL)

A compiler-level language used primarily for writing Message Control Systems (MCS) for B 1800/B 1700 Data Communications Systems.

Vertical Redundancy Checking (VRC)

Vertical redundancy checking consists of generating an odd-parity as each character is received. This technique is used only in the ASCII normal mode (not with EBCDIC). The test is performed on every character, including the LRC.

Voice-grade channel

A channel suitable for transmission of speech, digital, or analog data, or facsimile, generally with a frequency range of about 300 to 3000 Hertz.

Wait-before transmit positive acknowledgement (WACK)

The WACK character (used only for binary synchronous operation) allows a receiving station to indicate a "temporarily-not-ready-to-receive" condition to the transmitting station. This character can be sent as a response to a text or heading block, selection sequence (multipoint), line bid (point-to-point with contention), or as an ID (identification) line bid sequence (switched network). WACK is a positive acknowledgement to the received data block or to selection.

The normal transmitting station response to WACK is ENQ, but EOT and DLE EOT are also valid responses. When ENQ is received, the station will continue. The ability to receive WACK is mandatory for all binary synchronous stations, but the capability is optional.

The WACK character is represented by a DLE character followed by a hexadecimal 7C.

Wideband channel

A channel wider in bandwidth than a voice-grade channel.

LIST OF REQUIRED STATEMENTS

The required statement or statements for each section of NDL appear below. Appropriate remarks are also included.

<u>Section</u>	<u>Required Statement or Remarks</u>
DECLARATION	This section is optional. No statements required.
REQUEST	Dependent upon the network configuration.
CONTROL	Dependent upon the network configuration.
TERMINAL	<p>BUFFERSIZE Statement TERMINAL REQUEST statement</p> <p>A warning message will be given by the compiler if the following statements are not included: TERMINAL ADDRESS statement TERMINAL TYPE statement TRANSMISSION NUMBER statement</p>
STATION	<p>MYUSE statement STATION TERMINAL statement</p> <p>A warning message will be given by the compiler if the following statement is omitted. STATION ADDRESS statement</p>
LINE	<p>LINE ADDRESS statement LINE CONTROL statement LINE STATION statement</p>
FILE	FAMILY statement

NDL PROGRAM EXAMPLE

Figure D-1 illustrates the physical data communications network that is defined by the NDL program example in this appendix.

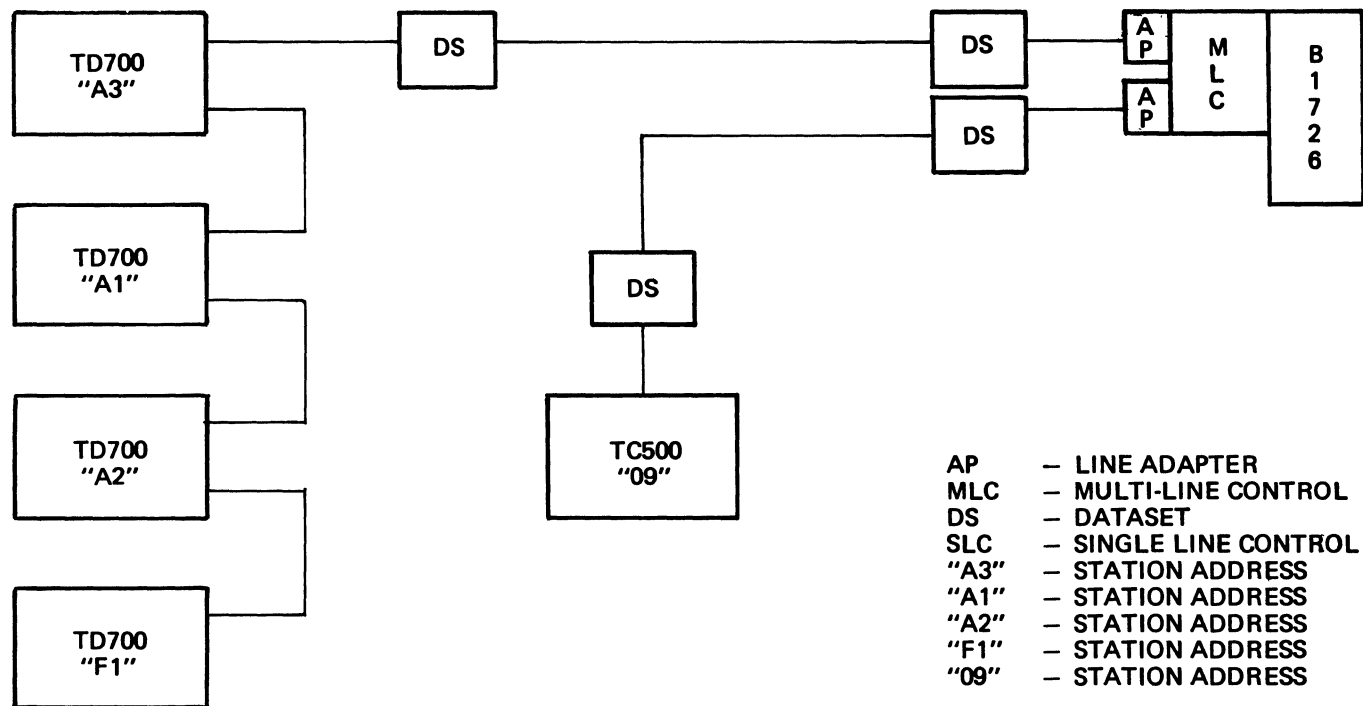


Figure D-1. Physical Definition of a Data Communications Network

Figure D-2 illustrates the logical relationship of the parts of the data communications network that is defined by the NDL program example in this appendix.

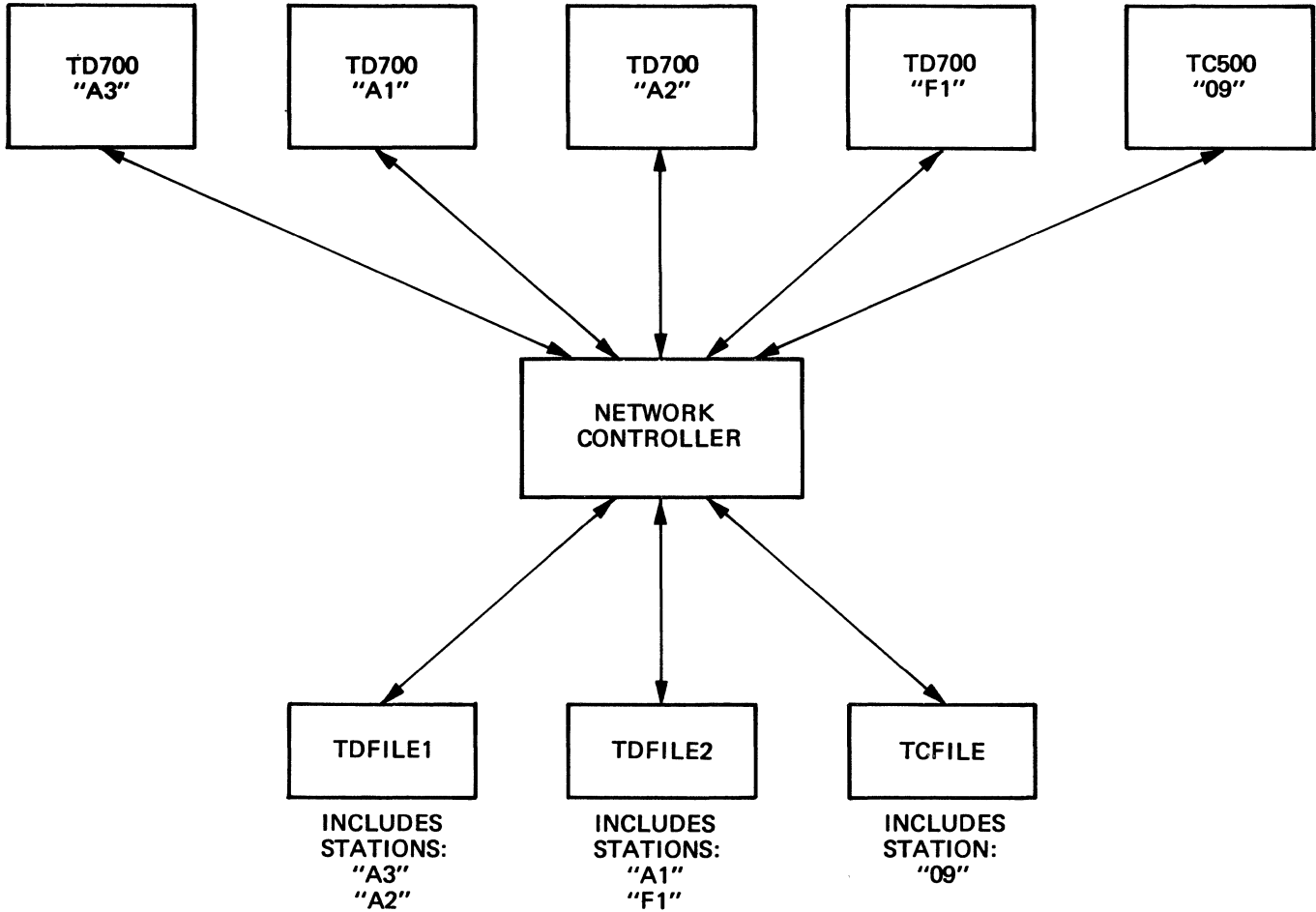


Figure D-2. Logical Definition of a Data Communications Network

DEMO/HANDLER COMPILE TO LIBRARY

```

SEQUENCE : SOURCE IMAGE %                               : IDENTIFIER SECTION
:DECLARATIONS: %                                       :
:   NIF = "DEMO"/"NIF". %                               :           DECLARATIO
$LIBRARY POLLTCTD
L   :REQUEST POLLTCTD:%                                   :           DECLARATIO
L   :%                                                    :           POLLTCTD REQUEST
L   :%                                                    :           POLLTCTD REQUEST
L   :%*****%                                           :           POLLTCTD REQUEST
L   :% * LIBRARY POLL REQUEST *                          :           POLLTCTD REQUEST
L   :%*****%                                           :           POLLTCTD REQUEST
L   :%                                                    :           POLLTCTD REQUEST
L   :%                                                    :           POLLTCTD REQUEST
L   :% USE "POLSELCTL", "AUTOPOLCTL"                     :           POLLTCTD REQUEST
L   :% LIBRARY LINE CONTROLS WITH THIS REQUEST          :           POLLTCTD REQUEST
L   :%                                                    :           POLLTCTD REQUEST
L   :% USE THIS REQUEST TO RECEIVE (POLL) FROM           :           POLLTCTD REQUEST
L   :% TC500, TC700, TC3500                               :           POLLTCTD REQUEST
L   :% TC4000, TC5000, TU500                              :           POLLTCTD REQUEST
L   :% TU700, TD700, TD800, TT142,                       :           POLLTCTD REQUEST
L   :% S1200                                               :           POLLTCTD REQUEST
L   :%                                                    :           POLLTCTD REQUEST
L   :% THIS REQUEST CONTAINS TRANSMISSION NUMBERS       :           POLLTCTD REQUEST
L   :%                                                    :           POLLTCTD REQUEST
L   :%                                                    :           POLLTCTD REQUEST
L   :% STATION(TOG[1])= TRUE - TRANSMISSION # ERROR     :           POLLTCTD REQUEST
L   :% LINE(TOG[1]) - TRUE - AUTOPOLL LINE               :           POLLTCTD REQUEST
L   :% LINE(TOG[5]) - TRUE - SWITCHED LINE              :           POLLTCTD REQUEST
L   :%% - FALSE - LEASED OR DIRECT LINE                  :           POLLTCTD REQUEST
L   :% STATION(TALLY[1]) - END OF BUFFER ERROR RETRY COUNT :           POLLTCTD REQUEST
L   :%                                                    :           POLLTCTD REQUEST
L   :%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX% :           POLLTCTD REQUEST
L   :%                                                    :           POLLTCTD REQUEST
L   :   TOG[1] := 0.%                                     RESET TRANERR TOG :           POLLTCTD REQUEST
L   :   DO POLLTCD.%                                     :           POLLTCTD REQUEST
L   :   IF NOT LINE(TOG[1]) THEN                          :           POLLTCTD REQUEST
L   :   DO. % NOT AUTCPOLL                                :           POLLTCTD REQUEST
L   :   INITIATE TRANSMIT.%                               :           POLLTCTD REQUEST
L   :   TRANSMIT EOT,ADDRESS(TRANSMIT),POL,ENG.          :           POLLTCTD REQUEST
L   :   FINISH TRANSMIT(NO EOT)-INITIATE RECEIVE.        :           POLLTCTD REQUEST
L   :   END.                                              :           POLLTCTD REQUEST
L   :   IF EXCEPTION THEN%                                :           POLLTCTD REQUEST
L   :   UNDO POLLTCD.%                                   :           POLLTCTD REQUEST
L   :   INITIALIZE TEXT.%                                :           POLLTCTD REQUEST
L   :   FETCH CHARACTER.%                                :           POLLTCTD REQUEST
L   :   IF CHARACTER EQ SOH THEN                          :           POLLTCTD REQUEST
L   :   DO SOH1.%                                         :           POLLTCTD REQUEST
L   :   RECEIVE ADDRESS(RECEIVE).                         :           POLLTCTD REQUEST
L   :   IF ADDERR THEN UNDO POLLTCD.%                    :           POLLTCTD REQUEST
L   :   RECEIVE TRAN(RECEIVE).                           :           POLLTCTD REQUEST
L   :   IF TRANERR THEN TOG[1] := 1.%SET STN TRAN TOG   :           POLLTCTD REQUEST

```

D-3

L	:	RECEIVE STX.%	:	POLLTCTD	REQUEST
L	:	IF FORMATERR THEN UNDO POLLTCD.	:	POLLTCTD	REQUEST
L	:	RECEIVE TEXT,ETX.%	:	POLLTCTD	REQUEST
L	:	IF FORMATERR THEN UNDO POLLTCD.	:	POLLTCTD	REQUEST
L	:	IF NOT TOG[1] THEN TERMINATE INPUT(RETURN,%	:	POLLTCTD	REQUEST
L	:	NO BUFFER).%	:	POLLTCTD	REQUEST
L	:	INITIATE TRANSMIT.%	:	POLLTCTD	REQUEST
L	:	TRANSMIT ACK.%	:	POLLTCTD	REQUEST
L	:	FINISH TRANSMIT-INITIATE RECEIVE.%	:	POLLTCTD	REQUEST
L	:	INITIALIZE RETRY.%	:	POLLTCTD	REQUEST
L	:	TERMINATE NOINPUT.%	:	POLLTCTD	REQUEST
L	:	END SCH1.%	:	POLLTCTD	REQUEST
L	:	IF CHARACTER EQ EOT THEN%	:	POLLTCTD	REQUEST
L	:	DO.%	:	POLLTCTD	REQUEST
L	:	INITIALIZE RETRY.%	:	POLLTCTD	REQUEST
L	:	TERMINATE NOINPUT.%	:	POLLTCTD	REQUEST
L	:	END.%	:	POLLTCTD	REQUEST
L	:	END PGLLTCD.%	:	POLLTCTD	REQUEST
L	:	%	:	POLLTCTD	REQUEST
L	:	IF LOSSOFDCSR THEN %	:	POLLTCTD	REQUEST
L	:	IF LINE(TOG[5]) THEN TERMINATE DISCONNECT.%	:	POLLTCTD	REQUEST
L	:	IF ENDOFBUFFER THEN%	:	POLLTCTD	REQUEST
L	:	DO.%	:	POLLTCTD	REQUEST
L	:	INITIALIZE TEXT.%	:	POLLTCTD	REQUEST
L	:	TALLY[1] := 20.%	:	POLLTCTD	REQUEST
L	:	TERMINATE INPUT(RETURN).%	:	POLLTCTD	REQUEST
L	:	DO EOBACK FOREVER.%	:	POLLTCTD	REQUEST
L	:	INITIATE TRANSMIT.%	:	POLLTCTD	REQUEST
L	:	TRANSMIT ACK.% PUT TERMINAL INTO REC	:	POLLTCTD	REQUEST
L	:	FINISH TRANSMIT-INITIATE RECEIVE.% RECEIVE EOT	:	POLLTCTD	REQUEST
L	:	IF NOT EXCEPTION THEN %	:	POLLTCTD	REQUEST
L	:	DO.%	:	POLLTCTD	REQUEST
L	:	INITIALIZE TEXT.%	:	POLLTCTD	REQUEST
L	:	FETCH CHARACTER.%	:	POLLTCTD	REQUEST
L	:	IF CHAR EQ EOT THEN UNDO EOBACK.%	:	POLLTCTD	REQUEST
L	:	END.%	:	POLLTCTD	REQUEST
L	:	TALLY[1] := TALLY[1]-1.%	:	POLLTCTD	REQUEST
L	:	IF TALLY[1] = 0 THEN UNDO EOBACK.%	:	POLLTCTD	REQUEST
L	:	END EOBACK.%	:	POLLTCTD	REQUEST
L	:	IF TRAN(RECEIVE) = "0" THEN TRAN(RECEIVE) := "1".%	:	POLLTCTD	REQUEST
L	:	ELSE TRAN(RECEIVE) := "0".%	:	POLLTCTD	REQUEST
L	:	TERMINATE OUTPUT(RETURN).%	:	POLLTCTD	REQUEST
L	:	INITIATE TRANSMIT.%	:	POLLTCTD	REQUEST
L	:	TRANSMIT EOT, ADDRESS(TRANSMIT), SEL, ENQ.%	:	POLLTCTD	REQUEST
L	:	FINISH TRANSMIT(NO EOT)-INITIATE RECEIVE.% RECEIVE ACK	:	POLLTCTD	REQUEST
L	:	INITIATE TRANSMIT.%	:	POLLTCTD	REQUEST
L	:	TRANSMIT SOF, ADDRESS(TRANSMIT).%	:	POLLTCTD	REQUEST
L	:	TRANSMIT TRAN(TRANSMIT), SIX.%	:	POLLTCTD	REQUEST
L	:	TRANSMIT "*" ABORT/XMIT: MSG GTR BUFFERSIZE *".%	:	POLLTCTD	REQUEST
L	:	TRANSMIT ETX.%	:	POLLTCTD	REQUEST
L	:	FINISH TRANSMIT-INITIATE RECEIVE.% RECEIVE ACK	:	POLLTCTD	REQUEST
L	:	IF TRAN(TRANSMIT) = "0" THEN TRAN(TRANSMIT) := "1".%	:	POLLTCTD	REQUEST
L	:	ELSE TRAN(TRANSMIT) := "0".%	:	POLLTCTD	REQUEST
L	:	TERMINATE NOINPUT.%	:	POLLTCTD	REQUEST


```

L      :          TRANSMIT ETX.%          : FASTSELTCO  REQUEST
L      :          END.%                  : FASTSELTCO  REQUEST
L      :          FINISH TRANSMIT(NO EOT)-INITIATE RECEIVE.% : FASTSELTCO  REQUEST
L      :          IF EXCEPTION THEN%      : FASTSELTCO  REQUEST
L      :              UNDO FSLTCD.%        : FASTSELTCO  REQUEST
L      :          INITIALIZE TEXT.%        : FASTSELTCO  REQUEST
L      :          FETCH CHARACTER.%        : FASTSELTCO  REQUEST
L      :          IF CHAR EQ ACK THEN%     : FASTSELTCO  REQUEST
L      :              DO.%                  : FASTSELTCO  REQUEST
L      :                  INITIALIZE RETRY.% : FASTSELTCO  REQUEST
L      :                  IF (TERMINALTYPE NE 42) AND % : FASTSELTCO  REQUEST
L      :                      (TERMINALTYPE NE 32) THEN % : FASTSELTCO  REQUEST
L      :                      INCREMENT TRAN(TRANSMIT). : FASTSELTCO  REQUEST
L      :                  ELSE%             : FASTSELTCO  REQUEST
L      :                      IF TRAN(TRANSMIT) EQ "0" : FASTSELTCO  REQUEST
L      :                      THEN TRAN(TRANSMIT) := "1". : FASTSELTCO  REQUEST
L      :                      ELSE TRAN(TRANSMIT) := "0". : FASTSELTCO  REQUEST
L      :                  TERMINATE OUTPUT.% : FASTSELTCO  REQUEST
L      :              END.%                 : FASTSELTCO  REQUEST
L      :          IF CHAR EQ NAK THEN %      : FASTSELTCO  REQUEST
L      :              DO.%                  : FASTSELTCO  REQUEST
L      :                  INITIALIZE RETRY.% : FASTSELTCO  REQUEST
L      :                  TERMINATE NOINPUT.% : FASTSELTCO  REQUEST
L      :              END.%                 : FASTSELTCO  REQUEST
L      :          END FSLTCD.%              : FASTSELTCO  REQUEST
L      :%                                     : FASTSELTCO  REQUEST
L      :          IF LOSSOFDSR AND NCT LINE(TOG[4]) THEN % : FASTSELTCO  REQUEST
L      :              IF LINE(TOG[5]) THEN TERMINATE DISCONNECT.% : FASTSELTCO  REQUEST
L      :          IF TIMEOUT THEN TERMINATE NOINPUT.% : FASTSELTCO  REQUEST
L      :          IF RETRY GT 0 THEN%        : FASTSELTCO  REQUEST
L      :              DO.%                  : FASTSELTCO  REQUEST
L      :                  RETRY:=RETRY-1.%   : FASTSELTCO  REQUEST
L      :                  TERMINATE NOINPUT.% : FASTSELTCO  REQUEST
L      :              END.%                 : FASTSELTCO  REQUEST
L      :          INITIALIZE RETRY. %         : FASTSELTCO  REQUEST
L      :          TERMINATE ERROR.%          : FASTSELTCO  REQUEST
L      :%                                     : FASTSELTCO  REQUEST
L      :%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX : FASTSELTCO  REQUEST
L      :%                                     : FASTSELTCO  REQUEST
$LIBRARY SELECTCTD
L      :%REQUEST SELECTCTD:%              : FASTSELTCO  REQUEST
L      :%                                     : SELECTCTD  REQUEST
L      :%                                     : SELECTCTD  REQUEST
L      :%                                     : SELECTCTD  REQUEST
L      :%                                     : SELECTCTD  REQUEST
L      :%                                     : SELECTCTD  REQUEST
L      :%                                     : SELECTCTD  REQUEST
L      :%                                     : SELECTCTD  REQUEST
L      :% USE "POLSELCTL", "AUTOPOLCTL", "AUTODYNCTL" : SELECTCTD  REQUEST
L      :% LIBRARY LINE CONTROLS WITH THIS REQUEST : SELECTCTD  REQUEST
L      :%                                     : SELECTCTD  REQUEST
L      :%                                     : SELECTCTD  REQUEST
L      :% USE THIS REQUEST TO TRANSMIT (SELECT) TO : SELECTCTD  REQUEST
L      :% TC500, TC700, TC3500 : SELECTCTD  REQUEST
L      :% TC4000, TC5000, TU500 : SELECTCTD  REQUEST
L      :% TC700, TD700, TD800, TI142 : SELECTCTD  REQUEST

```

```

L      :X      S1200                                :      SELECTCTD  REQUEST
L      :X      :                                  :      SELECTCTD  REQUEST
L      :X      THIS REQUEST CONTAINS TRANSMISSION NUMBERS :      SELECTCTD  REQUEST
L      :X      TERMINAL TYPE MUST BE SET.           :      SELECTCTD  REQUEST
L      :X      :                                  :      SELECTCTD  REQUEST
L      :X      :                                  :      SELECTCTD  REQUEST
L      :X      LINE(TOG(5))          - LINE OPERATION MODE. :      SELECTCTD  REQUEST
L      :X      0 - LEASED OR DIRECT                :      SELECTCTD  REQUEST
L      :X      1 - SWITCHED                       :      SELECTCTD  REQUEST
L      :XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX :      SELECTCTD  REQUEST
L      :X      :                                  :      SELECTCTD  REQUEST
L      :      DO SELECTTCD.%                        :      SELECTCTD  REQUEST
L      :      INITIATE TRANSMIT.%                   :      SELECTCTD  REQUEST
L      :      TRANSMIT EOT,ADDRESS(TRANSMIT),SEL,ENQ. :      SELECTCTD  REQUEST
L      :      FINISH TRANSMIT(NO EOT)-INITIATE RECEIVE.% :      SELECTCTD  REQUEST
L      :      IF EXCEPTION THEN%                   :      SELECTCTD  REQUEST
L      :      UNDO SELECTTCD.%                     :      SELECTCTD  REQUEST
L      :      INITIALIZE TEXT.%                    :      SELECTCTD  REQUEST
L      :      FETCH CHARACTER.%                    :      SELECTCTD  REQUEST
L      :      IF CHARACTER EQ ACK THEN%            :      SELECTCTD  REQUEST
L      :      DO REACK FOREVER.%                   :      SELECTCTD  REQUEST
L      :      INITIATE TRANSMIT.%                   :      SELECTCTD  REQUEST
L      :      TRANSMIT SOH,ADDRESS(TRANSMIT).      :      SELECTCTD  REQUEST
L      :      TRANSMIT TRAN(TRANSMIT).%           :      SELECTCTD  REQUEST
L      :      TRANSMIT STX,TEXT,ETX.%             :      SELECTCTD  REQUEST
L      :      IF ENDOFBUFFER THEN %               :      SELECTCTD  REQUEST
L      :      DO.%                                  :      SELECTCTD  REQUEST
L      :      INITIALIZE TEXT.%                    :      SELECTCTD  REQUEST
L      :      INITIATE TRANSMIT.%                   :      SELECTCTD  REQUEST
L      :      TRANSMIT SOH, ADDRESS(TRANSMIT).%    :      SELECTCTD  REQUEST
L      :      TRANSMIT TRAN(TRANSMIT),,STX.%      :      SELECTCTD  REQUEST
L      :      TRANSMIT  "* ABORT/RCV: MSG GTR ".%  :      SELECTCTD  REQUEST
L      :      TRANSMIT "BUFFERSIZE *".%           :      SELECTCTD  REQUEST
L      :      TRANSMIT ETX.%                       :      SELECTCTD  REQUEST
L      :      END.%                                :      SELECTCTD  REQUEST
L      :      FINISH TRANSMIT-INITIATE RECEIVE.%  :      SELECTCTD  REQUEST
L      :      IF EXCEPTION THEN%                   :      SELECTCTD  REQUEST
L      :      UNDO SELECTTCD.%                     :      SELECTCTD  REQUEST
L      :      INITIALIZE TEXT.%                    :      SELECTCTD  REQUEST
L      :      FETCH CHARACTER.%                    :      SELECTCTD  REQUEST
L      :      IF CHARACTER EQ ACK THEN%            :      SELECTCTD  REQUEST
L      :      DO.%                                  :      SELECTCTD  REQUEST
L      :      INITIALIZE RETRY.%                   :      SELECTCTD  REQUEST
L      :      IF (TERMINALTYPE NE 42) AND          :      SELECTCTD  REQUEST
L      :      (TERMINALTYPE NE 32) THEN %         :      SELECTCTD  REQUEST
L      :      INCREMENT TRAN(TRANSMIT).          :      SELECTCTD  REQUEST
L      :      ELSE%                                :      SELECTCTD  REQUEST
L      :      IF TRAN(TRANSMIT) EQ "0"           :      SELECTCTD  REQUEST
L      :      THEN TRAN(TRANSMIT) := "1".        :      SELECTCTD  REQUEST
L      :      ELSE TRAN(TRANSMIT) := "0".        :      SELECTCTD  REQUEST
L      :      TERMINATE OUTPUT.%                 :      SELECTCTD  REQUEST
L      :      END.%                                :      SELECTCTD  REQUEST
L      :      ELSE IF CHAR EQ NAK THEN %          :      SELECTCTD  REQUEST

```

```

L      :                               DO.%                               : SELECTCTD REQUEST
L      :                               IF RETRY = 0 THEN%                   : SELECTCTD REQUEST
L      :                               DO.%                               : SELECTCTD REQUEST
L      :                               INITIALIZE RETRY.%                   : SELECTCTD REQUEST
L      :                               TERMINATE ERROR.%                   : SELECTCTD REQUEST
L      :                               END.%                               : SELECTCTD REQUEST
L      :                               ELSE RETRY := RETRY-1.%             : SELECTCTD REQUEST
L      :                               END.%                               : SELECTCTD REQUEST
L      :                               ELSE UNDO SELECTTCD. %               : SELECTCTD REQUEST
L      :                               END RECAK.%                          : SELECTCTD REQUEST
L      :                               IF CHAR EQ NAK THEN %                : SELECTCTD REQUEST
L      :                               DG.%                               : SELECTCTD REQUEST
L      :                               INITIALIZE RETRY.%                   : SELECTCTD REQUEST
L      :                               TERMINATE NOINPUT.%                 : SELECTCTD REQUEST
L      :                               END.%                               : SELECTCTD REQUEST
L      :                               END SELECTTCD.%                     : SELECTCTD REQUEST
L      :%                               : SELECTCTD REQUEST
L      :                               IF LOSOFFDSR AND NOT LINE(TOG[4]) THEN % : SELECTCTD REQUEST
L      :                               IF LINE(TOG[5]) THEN TERMINATE DISCONNECT.% : SELECTCTD REQUEST
L      :                               IF TIMEOUT THEN TERMINATE NOINPUT.%   : SELECTCTD REQUEST
L      :                               IF RETRY GT 0 THEN%                   : SELECTCTD REQUEST
L      :                               DO.%                               : SELECTCTD REQUEST
L      :                               RETRY := RETRY-1.%                   : SELECTCTD REQUEST
L      :                               TERMINATE NOINPUT.%                 : SELECTCTD REQUEST
L      :                               END.%                               : SELECTCTD REQUEST
L      :                               INITIALIZE RETRY. %                   RESET RETRY COUNTER : SELECTCTD REQUEST
L      :                               TERMINATE ERROR. %                   TERMINATE ON ERROR  : SELECTCTD REQUEST
L      :%                               : SELECTCTD REQUEST
L      :%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX% : SELECTCTD REQUEST
L      :%                               : SELECTCTD REQUEST

```

\$PAGE

\$LIBRARY AUTOPOLCTL

```

L      : CONTROL AUTOPOLCTL :                               : SELECTCTD REQUEST
L      :%                               :                               : AUTOPOLCTL CONTROL
L      :%                               *****                               : AUTOPOLCTL CONTROL
L      :%                               * LIBRARY *                               : AUTOPOLCTL CONTROL
L      :%                               * AUTOPOLL CONTROL *                       : AUTOPOLCTL CONTROL
L      :%                               *****                               : AUTOPOLCTL CONTROL
L      :%                               : AUTOPOLCTL CONTROL
L      :%                               : AUTOPOLCTL CONTROL
L      :% USE LIBRARY REQUESTS                               : AUTOPOLCTL CONTROL
L      :% "POLLTCTD", "SELECTCTD", "FASTSELCTD"                 : AUTOPOLCTL CONTROL
L      :% WITH THIS LINE CONTROL                               : AUTOPOLCTL CONTROL
L      :%                               : AUTOPOLCTL CONTROL
L      :% LINE(TOG[0]) - SAVE LINE(TOG[1]) AUTOPOLL             : AUTOPOLCTL CONTROL
L      :% LINE(TOG[1]) - TRUE - AUTOPOLL STRING ALREADY CREATED : AUTOPOLCTL CONTROL
L      :% LINE(TOG[2]) - TRUE - CONTROL SELECT IF LINE(QUEUED) : AUTOPOLCTL CONTROL
L      :%                               - FALSE- CONTROL POLL         : AUTOPOLCTL CONTROL
L      :% LINE(TOG[3]) - TRUE - INITIATE IDLE IF CANNOT POLL OR SELECT : AUTOPOLCTL CONTROL
L      :% LINE(TOG[5]) - TRUE - SWITCHED LINE OPERATION         : AUTOPOLCTL CONTROL
L      :%                               - FALSE- LEASED OR DIRECT OPERATION : AUTOPOLCTL CONTROL
L      :% LINE(TALLY[0]) - SELECTED STATION NUMBER             : AUTOPOLCTL CONTROL

```

```

L      :% LINE(TALLY[1]) - POLLED STATION NUMBER : AUTOPOLCTL CONTROL
L      :% : AUTOPOLCTL CONTROL
L      :XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX : AUTOPOLCTL CONTROL
L      :% : AUTOPOLCTL CONTROL
L      :LINE(TOG[1]) := LINE(TOG[0]). : AUTOPOLCTL CONTROL
L      :% : AUTOPOLCTL CONTROL
L      :CASE LINE(CONTROL KEY). : AUTOPOLCTL CONTROL
L      :% : AUTOPOLCTL CONTROL
L      : . % C - PREVIOUS REQUEST COMPLETE : AUTOPOLCTL CONTROL
L      :% : AUTOPOLCTL CONTROL
L      : DO. % 1 - SYSTEM RECONFIGURATION : AUTOPOLCTL CONTROL
L      : LINE(TOG[0]) := 0. % SAVE LINE(TOG[1]) : AUTOPOLCTL CONTROL
L      : LINE(TOG[1]) := 0. % RECREATE AUTOPOLL STRING : AUTOPOLCTL CONTROL
L      : LINE(TALLY[0]) := 1. % : AUTOPOLCTL CONTROL
L      : LINE(TALLY[1]) := 1. % : AUTOPOLCTL CONTROL
L      : END. : AUTOPOLCTL CONTROL
L      :% : AUTOPOLCTL CONTROL
L      : IF LINE(QUEUED) THEN % 2 - CONTENTION, FIRST MESSAGE QUEUED : AUTOPOLCTL CONTROL
L      : INITIATE CANCEL.% : AUTOPOLCTL CONTROL
L      : ELSE % AUTOMATIC 30-SECOND WAKEUP : AUTOPOLCTL CONTROL
L      : CONTINUE.% : AUTOPOLCTL CONTROL
L      :% : AUTOPOLCTL CONTROL
L      : DC. % 3 - SWITCHED LINE ESTABLISHED : AUTOPOLCTL CONTROL
L      : LINE(TOG[0]) := 0.% SAVE LINE(TOG[1]) : AUTOPOLCTL CONTROL
L      : LINE(TOG[1]) := 0.% RECREATE AUTOPOLL STRING : AUTOPOLCTL CONTROL
L      : LINE(TOG[5]) := 1. % INDICATES SWITCHED LINE : AUTOPOLCTL CONTROL
L      : LINE(TALLY[0]) := 1.% : AUTOPOLCTL CONTROL
L      : LINE(TALLY[1]) := 1.% : AUTOPOLCTL CONTROL
L      : END. % : AUTOPOLCTL CONTROL
L      :% : AUTOPOLCTL CONTROL
L      : CONTINUE. % 4 - PREVIOUS TERMINATE OUTPUT(RETURN) : AUTOPOLCTL CONTROL
L      :% : AUTOPOLCTL CONTROL
L      : DO.% 5 - DIALOUT : AUTOPOLCTL CONTROL
L      : LINE(TOG[0]) := 0.% SAVE LINE(TOG[1]) : AUTOPOLCTL CONTROL
L      : LINE(TOG[1]) := 0.% RECREATE AUTOPOLL STRING : AUTOPOLCTL CONTROL
L      : LINE(TOG[4]) := 1.% DIAL-OUT FLAG : AUTOPOLCTL CONTROL
L      : LINE(TOG[5]) := 1.% INDICATES SWITCHED LINE : AUTOPOLCTL CONTROL
L      : LINE(TALLY[0]) := 1.% : AUTOPOLCTL CONTROL
L      : LINE(TALLY[1]) := 1.% : AUTOPOLCTL CONTROL
L      : END.% : AUTOPOLCTL CONTROL
L      :% : AUTOPOLCTL CONTROL
L      :END CASE. : AUTOPOLCTL CONTROL
L      :% : AUTOPOLCTL CONTROL
L      :LINE(TOG[3]) := 1. : AUTOPOLCTL CONTROL
L      :% : AUTOPOLCTL CONTROL
L      :IF LINE(QUEUED) % : AUTOPOLCTL CONTROL
L      :THEN DO WHICHREQ FOREVER. % : AUTOPOLCTL CONTROL
L      : DO LGUE. % : AUTOPOLCTL CONTROL
L      : IF NOT LINE(TOG[2]) % : AUTOPOLCTL CONTROL
L      : THEN DC. % MUST POLL FIRST : AUTOPOLCTL CONTROL
L      : LINE(TOG[3]) := 0. % SET TO COME BACK TO SELECT : AUTOPOLCTL CONTROL
L      :% IF CANNOT POLL : AUTOPOLCTL CONTROL
L      : UNCO LGUE. % : AUTOPOLCTL CONTROL
L      : END. % : AUTOPOLCTL CONTROL

```

```

L      : LINE(TOG[2]) := 0. % SET TO POLL NEXT TIME : AUTOPOLCTL CONTROL
L      : STATION := LINE(TALLY[0]). % SELECT STATION : AUTOPOLCTL CONTROL
L      : DO OUTSEL FOREVER. % : AUTOPOLCTL CONTROL
L      : IF STATION = 1 % : AUTOPOLCTL CONTROL
L      : THEN STATION := MAXSTATION. % LOOK AT LAST STATION : AUTOPOLCTL CONTROL
L      : ELSE STATION := STATION - 1. % LOOK AT NEXT STATION : AUTOPOLCTL CONTROL
L      : IF STATION(READY) % : AUTOPOLCTL CONTROL
L      : AND STATION(QUEUED) % : AUTOPOLCTL CONTROL
L      : AND STATION(MYUSE) NE 1 % NOT INPUT ONLY : AUTOPOLCTL CONTROL
L      : THEN DO. % : AUTOPOLCTL CONTROL
L      : LINE(TALLY[0]) := STATION. % SAVE : AUTOPOLCTL CONTROL
L      : STATION SELECTED : AUTOPOLCTL CONTROL
L      : INITIATE OUTPUT. % INITIATE OUTPUT REQUEST : AUTOPOLCTL CONTROL
L      : END. % : AUTOPOLCTL CONTROL
L      : IF STATION = LINE(TALLY[0]) % : AUTOPOLCTL CONTROL
L      : THEN DO FINISEL. % : AUTOPOLCTL CONTROL
L      : IF LINE(TOG[3]) % : AUTOPOLCTL CONTROL
L      : THEN UNDO LQUE. % TRY TO POLL : AUTOPOLCTL CONTROL
L      : IF LINE(TOG[5]) THEN INITIATE DISCONNECT.% : AUTOPOLCTL CONTROL
L      : ELSE INITIATE IDLE. % : AUTOPOLCTL CONTROL
L      : END FINISEL. % : AUTOPOLCTL CONTROL
L      : END OUTSEL. % : AUTOPOLCTL CONTROL
L      : END LQUE. % : AUTOPOLCTL CONTROL
L      : % : AUTOPOLCTL CONTROL
L      : LINE(TOG[2]) := 1. % SET TO SELECT NEXT TIME : AUTOPOLCTL CONTROL
L      : STATION := LINE(TALLY[1]). % POLL STATION : AUTOPOLCTL CONTROL
L      : DO INPOLL FOREVER. % : AUTOPOLCTL CONTROL
L      : IF STATION = 1 % : AUTOPOLCTL CONTROL
L      : THEN STATION := MAXSTATION. % LOOK AT LAST STATION : AUTOPOLCTL CONTROL
L      : ELSE STATION := STATION - 1. % LOOK AT NEXT STATION : AUTOPOLCTL CONTROL
L      : IF STATION(READY) % : AUTOPOLCTL CONTROL
L      : AND STATION(ENABLED) % : AUTOPOLCTL CONTROL
L      : AND STATION(MYUSE) NE 2 % NOT OUTPUT ONLY : AUTOPOLCTL CONTROL
L      : THEN DO. % : AUTOPOLCTL CONTROL
L      : LINE(TALLY[1]) := STATION. % SAVE POLLED STATION : AUTOPOLCTL CONTROL
L      : LINE(TOG[0]) := LINE(TOG[1]). % SAVE LINE(TOG[1]) : AUTOPOLCTL CONTROL
L      : LINE(TOG[1]) := 0. % : AUTOPOLCTL CONTROL
L      : INITIATE INPUT. % INITIATE INPUT REQUEST : AUTOPOLCTL CONTROL
L      : END. % : AUTOPOLCTL CONTROL
L      : IF STATION = LINE(TALLY[1]) % : AUTOPOLCTL CONTROL
L      : THEN UNDO INPOLL. % : AUTOPOLCTL CONTROL
L      : END INPOLL. % : AUTOPOLCTL CONTROL
L      : IF LINE(TOG[3]) OR NOT LINE(QUEUED) THEN % : AUTOPOLCTL CONTROL
L      : INITIATE IDLE.% : AUTOPOLCTL CONTROL
L      : END WHICHREG. % : AUTOPOLCTL CONTROL
L      : % : AUTOPOLCTL CONTROL
L      : IF LINE(TOG[1]) THEN % : AUTOPOLCTL CONTROL
L      : DO SET.% : AUTOPOLCTL CONTROL
L      : LINE(TOG[2]) := 1.% SET TO SELECT NEXT : AUTOPOLCTL CONTROL
L      : INITIATE AUTOPOLL.% : AUTOPOLCTL CONTROL
L      : END SET.% : AUTOPOLCTL CONTROL
L      : STATION := 1. : AUTOPOLCTL CONTROL
L      : DO NEXTSTN FOREVER. : AUTOPOLCTL CONTROL
L      : IF STATION(READY) % : AUTOPOLCTL CONTROL

```

105 4/1/77


```

:%                               : TDFILE1  FILE
:%                               : TDFILE1  FILE
:FILE TDFILE2: %                 : TDFILE1  FILE
:   FAMILY = TD2,TD4. %         : TDFILE2  FILE
:   RESIDENT = DISK. %         : TDFILE2  FILE
:%                               : TDFILE2  FILE
:%                               : TDFILE2  FILE
:FILE TCFILE: %                  : TDFILE2  FILE
:   FAMILY = TC1. %             : TCFILE   FILE
:   RESIDENT = DISK. %         : TCFILE   FILE
:%                               : TCFILE   FILE
:XXXXXXXXXXXXXXXXXXXXXXXXXXXXX  : TCFILE   FILE
:%                               : TCFILE   FILE
:FINI: %                         : TCFILE   FILE

```

***** COMPILATION COMPLETE

MACRO COMPILATION DATE: C1/18/77, 15:04:50

PROGRAM STATISTICS

```

MEMORY REQUIRED TO RUN:      42303 BITS PLUS MESSAGE-QUEUE SPACE
MEMORY REQUIRED FOR NIF TABLES:  13343 BITS
NUMBER LINES:                2
NUMBER STATIONS:              5
NUMBER TERMINALS:             3
NUMBER FILES:                 3
NAME STACK SIZE:              1 ENTRIES
CONTROL STACK SIZE:           15 ENTRIES
PROGRAM POINTER STACK SIZE:   25 ENTRIES
EVALUATION STACK SIZE:       20 ENTRIES
VALUE STACK SIZE:             29343 BITS
PROGRAM STATIC MEMORY:        32383 BITS
PROGRAM DYNAMIC MEMORY:       0 BITS

```

REQUEST PAGE: 9

SEGMENT	SIZE IN BITS
00	116
01	6588
02	3970
03	4861

CONTROL PAGE: 8

SEGMENT	SIZE IN BITS
00	66
01	6333

CCMPLE STATISTICS

```

NUMBER OF ERRORS DETECTED:      0
NUMBER OF WARNINGS:             0
NUMBER OF CARDS SCANNED:        543
NUMBER OF TOKENS SCANNED:       1969
ELAPSED TIME (NOT PRCESSOR TIME): 00:05:11.3

```

APPENDIX E

NDL LIBRARY

GENERAL

The Network Definition Language Library consists of procedures used in the operation of terminals that interface with B 1800/B 1700 series computers. The library is a source language library that contains the REQUESTS and CONTROLS necessary to handle common line disciplines for the terminal devices referenced herein. By utilizing the library, users should find it unnecessary to write their own REQUESTS and CONTROLS.

ACCESSING THE LIBRARY

The library of line disciplines described herein is a released product of the NDL system and is named NDL/LIBRARY.

The library consists of a number of groups of source statements coded in the Network Definition Language, each group constituting a procedure. These procedures fall into two classes: those called REQUESTS and those called CONTROLS.

Including the appropriate \$ LIBRARY statement in a compilation deck places any of these procedures, intact, into an NDL compile. For instance, \$LIBRARY POLLTCTD, when placed correctly in a compilation deck passed to the NDL Compiler, causes all source statements in the procedure named POLLTCTD from the NDL/LIBRARY file to be included in the source code passed to the compiler.

A typical compilation deck can appear as follows:

```
?COMPILE HANDLER WITH NDL TO LIBRARY;
?DATA CARDS
DECLARATION:
  NIF = "HANDLER"/"NIF".
?LIBRARY POLLTCTD
?LIBRARY SELECTCTD
?LIBRARY POLSELCTL

  < TERMINAL SECTION STATEMENTS >
  < STATION SECTION STATEMENTS >
  < LINE SECTION STATEMENTS >
  < FILE SECTION STATEMENTS >
FINI.
?END;
```

The Network Controller generated by such a compilation deck, when aided by an application program, communicates to a network composed of TC-, TD-, TT-, and AE-type terminals.

NOTE

The \$LIBRARY cards take the place of NDL executable statements in the deck. The compiler replaces these \$LIBRARY cards with actual source statements found in the NDL/LIBRARY file under their respective name (POLLTCTD, SELECTCTD, POLSELCTL).

NDL/LIBRARY FILE

The NDL/LIBRARY file contains REQUESTS and CONTROLS, each designed to be used with a certain class or type of terminal. These REQUESTS and CONTROLS are described in the following pages and are grouped according to the line discipline employed. The description indicates the name and type of line discipline for which each REQUEST and CONTROL is intended.

A detailed description of the action in a REQUEST or CONTROL is provided, which may be used in conjunction with a full listing of that REQUEST or CONTROL to obtain a more complete understanding of a particular line discipline and its implementation.

Requests

The purpose of a REQUEST is to perform a particular communication procedure with the assigned station. REQUESTS can be grouped by function as input, output, and input/output REQUESTS.

INPUT REQUESTS

An input REQUEST performs two basic functions:

- a. Queries a terminal for messages it may have READY to transmit.
- b. Receives messages and routes them to the remote file to which the terminal is attached.

Poll or Autopoll Request

The POLLTCTD REQUEST issues one poll (when AUTOPOLL is not in operation) to a station, and receives either a message or an EOT indicating no messages ready.

The LINE(TOG[1] toggle is set by the initiating CONTROL if AUTOPOLL was initiated; otherwise, it is reset. This toggle is used to avoid issuing another poll if AUTOPOLL was in process.

The POLLTCTD REQUEST code is exited if no message is received in response to the poll. If a message is received, then it is acknowledged, queued for the MCS, or passed on to the user program's remote file with which the assigned station is associated. The REQUEST is then exited.

I/O errors, other than ENDOFBUFFER, LOSSOFDSR, and TIMEOUT, cause the retry count to be decremented and the REQUEST exited. If the line is a switched line, timeouts cause the retry count to be decremented to allow a line disconnect when the retry count equals 0. Consecutive errors that cause the retry count to go to zero cause the REQUEST to perform a TERMINATE ERROR or a TERMINATE DISCONNECT statement. This exiting of the REQUEST on a leased line leaves the station READY and queues an error message for the MCS, if an MCS is present. If there is no MCS in the system, the station remains READY and a message is passed on to the user program's remote file which causes the EXCEPTION branch to be taken on the READ of the remote file.

If the user program attempts to receive a message which is too large for the Network Controller to communicate, the POLLTCTD REQUEST informs the intended sender with the following error message and then proceeds to the next Data Comm transfer:

ABORT/XMIT: MSG GTR BUFFERSIZE

The POLLTCTD REQUEST checks incoming messages for a transmission number if one is declared in the TERMINAL Section. If two consecutive messages have the same transmission number, the second message is discarded.

Dynamic Request

The POLTCTDDYN REQUEST is intended to be used only with the AUTODYNCTL library CONTROL to provide dynamic reconfiguration of the AUTOPOLL poll list of stations on a given line. Lines may have more than one line address (referred to as line groups). Refer to the discussion of the AUTODYNCTL CONTROL for further information.

The POLTCTDDYN REQUEST performs the same way as the POLLTCTD REQUEST when no exceptions occur on any station. When a polled station incurs a TIMEOUT exception, an indicator is checked to determine its status prior to the exception. If the terminal was never recognized as being on-line it is polled once again. If the station times out again, it is removed from the poll list. If the station was in use without error, it is single-polled once; if it times out again, it is removed from the poll list and marked to be retried in three minutes.

If the exception incurred is LOSSOFDSR and if the line was established as switched, the line is disconnected; otherwise, the exception is treated as any other non-timeout exception.

The effect of the special action taken on a TIMEOUT exception is that when a station is powered on, the Network Controller determines how many stations are responding and re-creates the AUTOPOLL poll list accordingly. When a station is powered off, the AUTOPOLL poll list is similarly re-created, including only those stations that are actively responding.

If a station has been marked as once active but now powered off and is to be retried in three minutes, the entire list of stations is retested for availability and the poll list is created accordingly. Criteria for acceptance to the new poll list are:

- a. The station has never been used by any line or is not in use by any line now, and
- b. The station does not timeout when polled.

The value of three minutes for RETRY is a constant set in the library code for the AUTODYNCTL CONTROL and may be altered by the user if desired.

If the user program attempts to receive a message which is too large for the Network Controller to communicate, the POLTCTDDYN REQUEST informs the intended sender with the following error message and then proceeds to the next Data Comm transfer:

```
***ABORT/XMIT:  MSG GTR BUFFERSIZE***.
```

The CANDE input REQUEST named CANDEPOLTD performs the same way as the standard POLLTCTD REQUEST with the following additions.

An input control system is implemented which determines whether or not a station is to be included in the AUTOPOLL poll string based on timeouts on poll. TOG[14] is the flag indicating that this station has retried the specified number of consecutive timeouts on poll and is to be deleted from the poll string. After a period of time (refer to CANDETDCTL for this value), all stations with TOG[14] TRUE are included in the poll string for purposes of again determining their input status. Refer to the CANDETDCTL REQUEST for the exact description of the manner in which they are included in the poll string.

Output messages that are sent to a station are sometimes not permitted to be received because the terminal is in LOCAL mode or the message has an unrecoverable error. An output control system prevents repeated retry attempts to send messages under those conditions, because such attempts are detrimental to the response time of the other stations. TOG[10] indicates to the CONTROL that this station is not receiving output; the CONTROL does not attempt output until this flag is off. The receipt of a ? in the first character position followed immediately by an ETX (of the input message from a station), resets TOG[10] for that station and allows output to be sent. Also, any input including valid CANDE commands enables the

output queue again. Failing either of these situations, an elapsed time period of eight seconds during which no activity occurred for this station causes TOG[10] to be reset for all stations on the line.

Scrolling is available to TD820/830 type terminals when TYPE is set correctly in the TERMINAL Section of the Network Controller. An input of ?+ enables scrolling, and ?- disables scrolling. Scrolling may also be controlled programmatically by setting the value of TALLY[0]. When TALLY[0] is set to 254 and TYPE is correct, scrolling is enabled. When TALLY[0] is set to 255, scrolling is disabled.

Scrolling of input is performed as follows:

- a. Clear to the end of the line.
- b. Move the cursor to home position.
- c. Leave the terminal in RECEIVE mode.
- d. Scroll up one line. This puts the top line on the bottom of the screen and brings the second line up to the top of the screen.
- e. Clear the top line.

Error recovery is the same as the error recovery procedure used with the standard POLLTCTD REQUEST. ENDOFBUFFER conditions are handled the same with slight modifications to accommodate CANDE. TIMEOUTS are handled by the dynamic input control system. If a station is marked INPUT DISABLED by the input control system, a TERMINATE ERROR statement is performed to inform CANDE that it must recover any user program active at this station.

If the terminal is being used only by CANDE (no other job executed through CANDE is attached to it), the CANDEPOLTD REQUEST maintains the cursor position as required by CANDE. If the terminal is being used by any other job, the cursor position is not maintained in any way.

Point-To-Point Requests

The TCTUPTRCV and TDPTRCV input REQUESTS establish communication on a master/slave basis. Initially, the TCTUPTRCV or TDPTRCV REQUEST waits (without TIMEOUT) for the station to bid for status. When the station bids, communication is established and the input message is passed on to the user program's remote file or queued for the MCS, if present. When the transaction is completed, master status must be re-established by the station for the next message.

When in an output REQUEST and trying to establish master status, the input REQUEST also bids for master status; the output REQUEST code is then exited and control is given to the input REQUEST. The input REQUEST then waits (without TIMEOUT) for the station to bid again for master status.

For the TCTUPTRCV REQUEST, I/O errors other than TIMEOUT cause the retry count to be decremented and the REQUEST exited. The TDPTRCV REQUEST operates similarly except that exception conditions, upon establishing communication, do not directly decrement the retry counter. Consecutive errors that cause both retry counts to go to zero cause both REQUESTS to perform a TERMINATE ERROR or a TERMINATE DISCONNECT statement. This exit from the REQUEST on a leased line leaves the station READY and queues an error message for the MCS, if present. If there is no MCS in the system, the station remains READY and a message is passed to the user program's remote file, which causes the EXCEPTION branch to be taken on the READ of the remote file.

If a message received by a user program exceeds the BUFFERSIZE as defined in the Network Controller, the partial message which does not exceed the BUFFERSIZE is passed on to the intended receiver as if no errors had occurred.

Teletype † Request

The READTTY input REQUEST issues a read with no timeout, and upon receipt of ENQ, CR, or ETX, the carriage is automatically positioned to the left margin and advanced one line. Since the READTTY REQUEST is sensitive to a Carriage Return, an ETX, or an ENQ (as ending control codes), it must check which code was actually received before returning the carriage. If an ENQ is received, the message is discarded and a READ with NO TIMEOUT is initiated. If an ETX or CR is received, then the message is queued for the user program or MCS and the REQUEST is exited.

Any error, whether an I/O exception or receipt of an invalid response, causes the retry count to be decremented and the READTTY REQUEST exited. Consecutive errors which cause the retry count to go to zero cause the READTTY REQUEST to perform a TERMINATE ERROR or TERMINATE DISCONNECT statement which notifies the user program through the EXCEPTION branch of the READ. If an MCS is present, an error message is queued for it.

Attempts to transmit message from the Teletype keyboard which exceed the BUFFERSIZE defined in the Network Controller are aborted. The message:

```
***ABORT/XMIT: MSG GTR BUFFERSIZE***
```

is displayed on the Teletype console to indicate this error.

OUTPUT REQUESTS

Output REQUESTs perform the function of transmitting from the system to the terminal any message designated as output.

Select Requests

The basic purpose of the SELECTCTD REQUEST is to transmit messages to an assigned station by means of the standard select line discipline.

If an ACK is received in response to the select, the output message is sent. When the acknowledgement is received for the message, the transmission number is updated, the output message is cleared from memory and the SELECTCTD REQUEST is exited. If a NAK is received in response to the select, the output message is requeued and the SELECTCTD REQUEST is exited.

I/O errors other than ENDOFBUFFER, LOSSOFDSR, and TIMEOUT cause the RETRY count to be decremented and the SELECTCTD REQUEST exited. Timeouts on a switched line cause the retry counters to be decremented to eventually allow a line disconnect when the retry count equals zero. Consecutive errors that cause the retry count to go to zero cause the SELECTCTD REQUEST to perform a TERMINATE ERROR or TERMINATE DISCONNECT statement. If an MCS is present, this exiting of the SELECTCTD REQUEST on a leased line leaves the assigned station READY and queues an error message for the MCS. If there is no MCS in the system, the assigned station remains READY and a message is passed to the user program's remote file, which causes the EXCEPTION branch to be taken on the next READ of the remote file.

The SELECTCTD REQUEST sends a transmission number with the output message if TRANSMISSION is declared in the TERMINAL Section for the assigned station. The TC terminals (TC500 and TC700 for example) transmission numbers are incremented consecutively (i.e., 0, 1, 2, 3, etc.) whereas the TD terminals (TD700 and TD800 for example) alternate transmission numbers between 0 and 1.

If the user program attempts to send a message which is too large for the Network Controller, the SELECTCTD REQUEST notifies the intended receiver of the message by the following error notice and then proceeds to the next Data Comm transfer:

```
* ABORT/RCV: MSG GTR BUFFERSIZE *
```

† Teletype is a registered trademark of Teletype Corporation.

Dynamic Request

SELTCTDDYN is a special purpose REQUEST that is intended to be used only with the AUTODYNCTL CONTROL and in conjunction with input REQUEST POLTCTDDYN to provide dynamic reconfiguration of stations on a line. The SELTCTDDYN REQUEST performs the same way as the SELECTCTD REQUEST when communicating with a TD830 Series terminal. If TERMINALTYPE is not set to a value of 45 or 46, then the SELTCTDDYN REQUEST attempts a FASTSELECT. Upon an EXCEPTION or FORMAT error occurring on the READ of the FASTSELECT, the REQUEST sets TOG[3] to 1 and does a normal select of the station until the station is READY to receive. If TOG[3] is true and TERMINALTYPE equals 45 or 46, the SELTCTDDYN REQUEST always does a normal select as does the SELECTCTD REQUEST.

NOTE

Timeouts do not decrement the retry counter unless the line has been established as switched, thereby allowing disconnect when the retry count equals 0. If LOSSOFDSR is detected and the line is switched, immediate disconnect is performed on the line; otherwise, the retry counter is decremented.

The SELTCTDDYN REQUEST sends a transmission number with the output message if a transmission number is declared in the TERMINAL Section for the assigned station. The TC terminals (TC500 and TC700 for example) transmission numbers are incremented consecutively (i.e., 0, 1, 2, 3, etc.) whereas the TD terminals (TD700 and TD800 for example) alternate transmission numbers between 0 and 1.

If the user program attempts to send a message which is too large for the Network Controller to communicate, the SELTCTDDYN REQUEST notifies the intended receiver of the message by the following error message, and then proceeds to the next Data Comm transfer:

ABORT/RECV: MSG GTR BUFFERSIZE

The CANDESELTD REQUEST performs the same as the standard select REQUEST with the same additions as discussed in the FASTSELECT CANDE REQUEST. Refer to the CANDEFSLTD REQUEST for additional information.

In addition, the CANDESELTD REQUEST discards a message to a station when the station responds with an ACK to SELECT, but responds with a NAK to TEXT (retry consecutive times). This prevents messages containing irrecoverable errors in the text from degrading the system response time.

Fast Select Requests

The FASTSELTC D REQUEST functions similarly to the select REQUESTs except for the following differences:

- a. Instead of issuing one I/O to select the terminal and receive a response (ACK or NAK) and then issuing another I/O to transmit the output message, a single I/O is issued which notifies the destination terminal that an output message is enroute; immediately following the control sequence is the message itself. The terminal then returns ACK or NAK indicating receipt or non-receipt of the message.
- b. Error recovery is the same as that in the select REQUESTs.
- c. Messages which are too large for the Network Controller to communicate are handled as in the SELECTCTD REQUEST.

The CANDEFSLTD output REQUEST performs the same as the standard FASTSELECT REQUEST with the following additions:

- a. The output control system sets a timer and sets TOG[10] for a station which times out or responds with a NAK to FASTSELECT.
- b. Scrolling, when enabled, is performed on any output. The operator must:
 1. Move the cursor to home position.
 2. If less than 23 consecutive transmissions have occurred, leave the terminal in RECEIVE mode.
 3. Delete the top line.
 4. Move the cursor up one line. This places the cursor at the left, on the bottom line of the screen.
 5. Transmit the text.
 6. Move the cursor to home position.
- c. If the terminal that is to receive the message is defined as a TD830 series (TYPE equals 45 or 46), a SELECT is performed before the FASTSELECT occurs. If the response is an ACK, the REQUEST proceeds as usual; otherwise, the error routine is executed. Also, after a successful transmission to a TD830 series terminal, without any intervening input from that station, the next transmission to that station is given four retries in addition to the normal two allowed all stations. This additional retry factor is reset if any message is received; however, the next output, if successful, reactivates the additional retry factor.
- d. After two consecutive timeouts on a FASTSELECT, the station is marked as temporarily disabled for output and is retried after eight seconds.
- e. If there are one or more lines of output from some program other than CANDE currently on the screen of a terminal of a station with scrolling enabled, the station is placed in LOCAL mode before a page of output from CANDE is displayed.

Point-To-Point Request

The TCTUPTXMIT and TDPTXMIT output REQUESTs establish communication on a master/slave basis. The TCTUPTXMIT and TDPTXMIT REQUESTs initially attempt to obtain master status when an output message is queued. If these two REQUESTs are bidding for master status and the remote station is likewise bidding for status, the output REQUEST is exited, allowing the remote station to again bid for master status. Should the station fail to rebid for status within the adapter timeout period, the output REQUEST again attempts to establish master status before sending its queued message.

For the TCTUPTXMIT REQUEST, I/O errors other than TIMEOUT cause the retry count to be decremented and the REQUEST exited. However, timeouts on a switched line cause the retry counter to be decremented to allow a line disconnect when the retry equals 0.

The TDPTXMIT REQUEST operates the same except all I/O errors cause the retry counter to be decremented. Consecutive errors that cause the retry count to go to zero cause the REQUEST to perform a TERMINATE ERROR or a TERMINATE DISCONNECT statement. Such an exit of the REQUEST on a leased line leaves the assigned station READY and queues an error message for the MCS, if present. If there is no MCS in the system, the assigned station remains READY and a message is passed to the user program's remote file. This causes the EXCEPTION branch to be taken on the next READ of the remote file.

If the user program attempts to send a message which is too large for the Network Controller to communicate, the REQUEST notifies the intended receiver of the message with the following error message and then proceeds to the next Data Comm transfer:

ABORT/RECV: MSG GTR BUFFERSIZE

The TDBATCHXMT output REQUEST establishes communication on a master/slave basis. The TDBATCHXMIT REQUEST initially attempts to obtain master status upon having an output message queued. If upon bidding for master status the remote station is likewise bidding for status, the output REQUEST is exited allowing the remote station to again bid for master status. If the station fails to rebid for status within the adapter's timeout period, the output REQUEST again attempts to establish master status before transmitting its queued message. Upon transmitting its message, a check is made to determine whether or not other output messages are queued. If other output messages are queued, the REQUEST continues to transmit messages until the queue is empty; otherwise, the REQUEST is exited immediately.

I/O errors other than TIMEOUT and ENDOFBUFFER cause the retry count to be decremented and the TDBATCHXMT REQUEST exited; however, timeouts on a switched line cause the retry counter to be decremented to allow a line disconnect when the retry count equals 0. Consecutive errors that cause the retry count to go to zero cause the REQUEST to perform a TERMINATE ERROR or TERMINATE DISCONNECT statement. Such an exit of the REQUEST on leased lines leaves the assigned station READY and queues an error for the MCS, if present. If there is no MCS in the system, the assigned station remains READY and a message is passed to the user program's remote file. This causes the EXCEPTION branch to be taken on the next READ of the remote file.

If the user program attempts to send a message which is too large for the Network Controller to communicate, this REQUEST notifies the intended receiver of the message with the following error message and then proceeds to the next Data Comm transfer:

ABORT/RECV: MSG GTR BUFFERSIZE

Teletype Request

The WRITETTY REQUEST transmits every message appended with a carriage return and a line feed character. If the WRITE operation is completed without error, the output message is deallocated and the WRITETTY REQUEST is exited.

Any error except a BREAK causes the retry count to be decremented, the message requeued, and the REQUEST exited. Consecutive errors cause the REQUEST to perform a TERMINATE ERROR or a TERMINATE DISCONNECT statement, requeue the message, report the error to the MCS or the user program, and exit the REQUEST. If the error was a BREAK during the WRITE operation, the REQUEST automatically positions the carriage to the left, advances one line, and exits the REQUEST after deallocating the output message.

Attempts to transmit messages to the Teletype keyboard which exceed the BUFFERSIZE defined in the Network Controller are aborted. The following message is displayed on the Teletype console to indicate this error:

ABORT/RECV: MSG GTR BUFFERSIZE

INPUT/OUTPUT REQUESTS

An input/output REQUEST performs the functions of both input and output without the need to be initiated by the CONTROL more than once, except under certain error conditions.

Poll/Select Request

The purpose of the DIAGTCTDIO REQUEST is to provide a minimal diagnostic capability in a readily available form in the LIBRARY.

The DIAGTCTDIO REQUEST performs both input and output functions for multi-point terminals and provides a trace of all activity between the system and the terminal. The trace uses a AUDITFILE named DIAG which must be declared in the DECLARATION Section of any Network Controller which includes this REQUEST from the LIBRARY.

Example:

```
AUDITFILE DIAG (DEVICE = PRINTER OR BACKUP).
```

The operations of the input and output sections of this REQUEST are identical to those of the poll and select REQUESTs (POLLTCTD and SELECTCTD), respectively.

The diagnostic REQUEST can be specified in a non-MCS configuration in the TERMINAL Section as:

```
REQUEST = DIAGTCTDIO: RECEIVE,DIAGTCTDIO:TRANSMIT.
```

The diagnostic REQUEST is also an optional statement in a system utilizing an MCS, by specifying the following in the TERMINAL Section:

```
DIAGNOSTIC = DIAGTCTDIO: RECEIVE, DIAGTCTDIO: TRANSMIT
```

This option allows, through a DCWRITE from the MCS, the ability to exit the current REQUEST, enter the diagnostic REQUEST, and return to the original REQUEST if desired.

Point-To-Point Request

The TCTUPTIO REQUEST establishes communication on a master/slave basis. Initially the TCTUPTIO REQUEST waits without TIMEOUT for either an output message to be queued (which then cancels the TIMEOUT), or for a station to request permission to transmit its message buffer. Upon acknowledging the station's request, its message buffer is received and passed to the user program's remote file, or queued for the MCS, if present.

All output operations (for example, TRANSMIT ACK, TRANSMIT NAK, TRANSMIT TEXT) are determined by the value placed in LINE (TALLY[1] which is constantly being updated by the REQUEST according to I/O communication.

Any error, except ENDOFBUFFER, whether an I/O error or receipt of an incorrect message or response, causes the retry count to be decremented and the REQUEST exited. Consecutive errors that cause the retry count to go to zero cause the REQUEST to perform a TERMINATE ERROR or a TERMINATE DISCONNECT statement. Such an exit from the REQUEST on leased lines leaves the assigned station READY and queues an error message for the MCS, if present. If there is no MCS in the system, the assigned station remains READY and a message is passed to the user program's remote file. This causes the EXCEPTION branch to be taken on the next READ of the remote file.

The TCTUPTIO REQUEST should be used only with the CONVERCTL line CONTROL with STATION(NYUSE) declared as INPUT/OUTPUT.

Teletype Request

The CANDEIOTTY is an input/output conversational POINT-TO-POINT Teletype REQUEST. CANDEIOTTY performs similarly to the standard Teletype REQUESTs with the following additions:

- a. This REQUEST is sensitive to carriage return (CR) as an ending code on input.
- b. If a BREAK is detected during a WRITE operation, the REQUEST transmits a BREAK to the terminal. A flag is set to indicate to CANDE that a BREAK was requested.
- c. If an ENQ is sent, the REQUEST transmits a DEL to the terminal to indicate that the last keyed-in message was discarded.
- d. Error recovery is similar to that of the other CANDE REQUESTs.
- e. Carriage position is controlled by the needs of CANDE when the terminal is attached only to CANDE; otherwise, carriage position is maintained at the left margin on a unused line for the next input or output.

Remote Job Entry (RJE) Request

The RJE REQUEST consists of establishment and message transfer phases. During the establishment phase, ENQ is transmitted and ACK is expected in return. If ACK is received, the message transfer phase is entered. If NAK or no response occurs, the ENQ is reissued. If ENQ is received, this indicates that the remote station is also bidding for establishment; otherwise, the B 1800/B 1700 system transmits ACK and enters the message transfer phase.

Basic Line Discipline. The message transfer phase of the RJE REQUEST consists of a conversational procedure and the addition of another transmission number in each message to be used for responding to messages received. A conversational procedure is one in which either the host or remote system may receive a message in response to the transmission of a message. The received message must contain some response to the message just transmitted. This is handled by the additional transmission number in the following way:

The first transmission number (TR1) received is the response to the previous transmission. The second transmission number (TR2) is the number of the message being received.

If TR1 is equal to the transmission number of the last message transmitted, it implies acknowledgement of valid receipt of that message and the transmitted message may be discarded. If TR1 is not equal to the last message number, it implies that the message was not received and must be retransmitted. This is termed an implied NAK. The remainder of the message is then received, and if no errors occur the next message sent contains the previous message's TR2 and TR1 implying valid receipt of the message.

If the message was not received validly and we assume that the message was received from the host system, TR1 of the next message being sent contains the transmission number of the last validly received message from the host. This implies that the host must retransmit its current message.

If there is no message ready to be sent, TRn.ACK is sent implying valid receipt of the host system's last transmission and indicating that no messages are ready to send. Similarly, if the last message sent by the host system was not received validly but no messages are ready to be sent, then TRn.ACK is sent (TRn is the transmission number of the last validly received message). This implies that the host system must retransmit its current message.

If no messages are queued for transmission the REQUEST issues a READ, and if a TIMEOUT occurs, another READ is issued.

The procedures just described constitute the main loop of the RJE REQUEST and are in use the majority of the time during a normal run.

Error Recovery. The message transfer phase is exited under any of the following conditions:

- a. If the MCS requests re-establishment by setting TOG[1] in a message header, the messages queued for the current station which have TOG[0] are set (meaning the message has been retransmitted the number of times specified by RETRY) are discarded and the establishment phase is entered.
- b. Upon detecting LOSSOFDSR, the line is disconnected and a TEST-AND-WAIT for DATA-SET-READY is initiated on the line. When DATA-SET-READY is detected, the establishment phase is entered.
- c. Upon receipt from the MCS of a message with TOG[2] set in the message header, the line is disconnected as in item b. This convention is used to disconnect the line under MCS control.
- d. If, while looking for output messages, it is discovered that there are no stations ENABLED or QUEUED, the line is disconnected if it is a switched line, or idled if it is leased. Line (TOG[6]) is used for this purpose.
- e. Retries are implemented using the QUEUE OUTPUT construct which places the current output buffer on the top of the output queue for the current station. Re-queuing of the output buffer occurs only after transmission has been attempted the number of times specified by RETRY in the STATION Section.

File Control. If a user program opens a remote file OUTPUT ONLY, stations in that file are not ENABLED. INITIATE INPUTOUTPUT must be executed in order to obtain an output buffer, (implicit checking for STATION(ENABLED) is no longer performed by the Skeletal Network Controller). This condition only occurs after the REQUEST has initiated a READ and a system reconfiguration occurs, such as a file OPEN or CLOSE. The effect is that even though the REQUEST is conversational, implying both input and output occurring simultaneously, it may be used for output only if desired.

Variables. The toggles and tallies used in the RJE REQUEST and CONTROL are described at the beginning of the REQUEST.

Transmission number handling is completely transparent to the user.

RETRY is a variable set in the STATION Section and may have a value of 0 to 255 inclusive.

RJE/HOST Request

The RJE/HOST REQUEST is similar to the RJE REQUEST, but has the ability to communicate to the MCS. By assigning a value to TALLY[0] before sending a message to the MCS (which can be a dummy message), the Network Controller can inform the MCS of any status changes concerning any of the stations accessed by the REQUEST. The valid settings for TALLY[0] are:

<u>Value</u>	<u>Meaning</u>
0	LOSS OF DSR detected
1	“DLE-EOT” either sent or received
2	“DLE-ENQ” received
3	“ACK” received
4	Message received
5	DIALOUT failed

Values 0 and 1 inform the MCS that the current station has disconnected and should be removed from the system. Values 2 and 3 inform the MCS that the current station has established connection and is entering the RJE system. Value 4 flags the message as a data message received from the station. Values 0 through 3 indicate dummy stations.

In addition, the RJE/HOST REQUEST has the ability to go into a READ NO TIMEOUT state when a particular station fails to establish a connection. After 100 attempts to establish a connection, the Network Controller goes into a READ NO TIMEOUT situation until a user becomes active on that station if the line is a leased line; otherwise, it is disconnected.

The RJE/HOST REQUEST also contains a mechanism which, if a line becomes established and after 100 reads no indication of activity is seen, disconnects the line. A message received with TOG[3] set turns this mechanism off, while a message received with TOG[4] set or a termination of the connection cause the mechanism to be turned on. LINE(TALLY[5]) is used as the counter for this retry mechanism.

The RJE/HOST REQUEST also supports DIALOUT. LINE(TOG[7]) is used to indicate a DIALOUT state. Should DIALOUT fail, a dummy message is queued for the MCS.

LINE(TOG[8]) is used as an indication that a termination REQUEST was received from the MCS.

Controls

The purpose of a CONTROL is to assign a station for activity and initiate the proper REQUEST to perform a communication procedure.

MULTI-POINT CONTROLS

A multi-point CONTROL can handle a line with more than one station attached.

Poll/Select Control

The basic purpose of the POLSELCTL CONTROL is to initiate the appropriate input (poll) or output (select) REQUEST after having assigned a station to the line that is ready to communicate. LINE(TOG[1]) is set false indicating that autopoll is not being initiated.

Upon completion of a previous REQUEST, when LINE(CONTROL KEY) equals 0, or upon notification of system reconfiguration, when LINE(CONTROL KEY) equals 0, the CONTROL checks each station on the line starting at the next station. If the station is READY, QUEUED, and MYUSE is not INPUT ONLY, the output REQUEST is initiated. If the station is READY, ENABLED, and MYUSE is not OUTPUT only, the input REQUEST is initiated. The failure to find any station on the line whose REQUEST can be initiated causes the line to be put in the idle state. LINE(TALLY[1]) is used to determine when all the stations on the line have been checked.

When the line has a READ with NO TIMEOUT pending and a message is queued for the line where there was no prior message in the queue, the CONTROL is entered with LINE(CONTROL KEY) equal to 2 and the CONTROL cancels the I/O in process. The CONTROL is then re-entered with LINE(CONTROL KEY) equal to 0.

If the station establishes itself on a switched line, the CONTROL is entered with LINE(CONTROL KEY) equal to 3 or LINE(CONTROL KEY) equal to 5 if an ACU is used; the CONTROL sets LINE(TOG[5]) equal to 1. Upon failure to initiate any REQUEST for the station, if LINE(TOG[5]) equals 1 and the line is queued, the line is disconnected; otherwise, it is put into the idle state.

If a REQUEST performs a TERMINATE OUTPUT(RETURN) statement, the CONTROL is entered with LINE(CONTROL KEY) equal to 4. The CONTROL performs a CONTINUE statement which assigns an output message buffer to the line for the same station, if a message is queued, and returns to the REQUEST from which the TERMINATE OUTPUT(RETURN) was executed.

LINE(TOG[4]) is provided for the convenience of the user program for flagging an ACU line. This toggle is not set or reset in the REQUESTs or CONTROLs.

Autopoll Controls

The basic purpose of the AUTOPOLCTL CONTROL is to initiate the appropriate input (autopoll) or output (select) REQUEST after having assigned a station to the line that is ready to communicate. LINE(TOG[1]) is set true if the line is to be automatically polled and if the autopoll string exists. LINE(TOG[0]) is used to save the contents of LINE(TOG[1]) indicating that the poll string has already been created when a normal poll is to be initiated.

Upon completion of a previous REQUEST, LINE(CONTROL KEY) equals 0, or upon notification of system reconfiguration, LINE(CONTROL KEY) equals 1 (which also sets LINE(TOG[0]) and LINE(TOG[1]) false, thereby indicating the autopoll string is to be recreated), the CONTROL determines if the line has any messages queued for any of its stations. If there are messages queued, the CONTROL checks each station on the line starting with the next station. If the station is READY, QUEUED, and MYUSE is not INPUT only, the output REQUEST is initiated. If the station is READY, ENABLED, and MYUSE is not OUTPUT only, LINE(TOG[1]) is saved and set false to indicate no autopoll, and the input REQUEST is initiated. LINE(TALLY[1]) is used to determine how many stations have been checked.

If there are no messages queued for output on the line and LINE(TOG[1]) is true, AUTOPOLL is initiated. If LINE(TOG[1]) is false, an autopoll string is created for the line. Each station on the line that is READY, ENABLED, and MYUSE is not OUTPUT only, is put into the autopoll string until all the stations are checked, or the autopoll buffer, defined in the LINE Section, is filled. If at least one station was put into the autopoll string, LINE(TOG[1]) and LINE(TOG[0]) are set true. The AUTOPOLCTL CONTROL allows a station to appear only once in the autopoll string. If LINE(TOG[1]) is true AUTOPOLL is initiated. If LINE(TOG[1]) is false and LINE(TOG[5]) is true, the line is disconnected; otherwise, it is put into the idle state.

When the line has an autopoll I/O in process and a message is queued for the line where there was no prior message in the queue, the CONTROL is entered with LINE(CONTROL KEY) equal to 2 and the CONTROL cancels the I/O in process. The CONTROL is then re-entered with LINE(CONTROL KEY) equal to 0.

If the autopoll I/O continues in process without any output message being initiated at the terminals for a period of 30 seconds, the Skeletal Network Controller automatically enters the CONTROL with LINE(CONTROL KEY) set to a value of 2. The CONTROL determines that no message is queued and does not cancel the I/O, allowing it to continue as before. The purpose of this automatic wake-up is to allow a CONTROL to cancel an I/O that may continue forever if no terminal or system activity occurs to interrupt it. AUTOPOLL and READ with NO TIMEOUT are two such I/O's. Refer to the description for an implementation of this feature.

If the station establishes itself on a switched line, the CONTROL is entered with LINE(CONTROL KEY) equal to 3 or with LINE(CONTROL KEY) is equal to 5 if the line is a Automatic Calling Unit (ACU), at which time the CONTROL sets LINE(TOG[5]) is equal to 1. Upon failure to initiate any REQUEST for the station, if LINE(TOG[5]) is equal to 1 and the line is queued, the line automatically disconnects; otherwise, it is put into the idle state.

If a REQUEST performs a TERMINATE OUTPUT(RETURN) statement, the CONTROL is entered with LINE(CONTROL KEY) equal to 4. The CONTROL performs a CONTINUE statement which lets an output message buffer be assigned to the line for the same station if one is queued, and returns to the REQUEST from which the TERMINATE OUTPUT(RETURN) statement was executed.

Dynamic Control

AUTODYNCTL is a special CONTROL intended to provide dynamic reconfiguration of the autopoll poll list when stations defined on a line are not responding, or when any of a defined set of stations may become attached to any of the lines of a group, in switched operation on dial-in lines.

The AUTODYNCTL CONTROL performs four basic functions:

- a. Determines the responding stations on a line by single-polling each station.
- b. Builds a poll list containing only the responding stations and initiates AUTOPOLL for the line.
- c. If any station was found not responding, initiates and maintains a three-minute timer which performs items (a) and (b) again.
- d. Initiates output to a station as necessary if the station is READY and VALID (TALLY[LINE] equals 0 or 4).

If no stations are found ENABLED, the CONTROL resets TALLY[LINE] to 0.

The AUTODYNCTL CONTROL has the following advantages over the AUTOPOLL CONTROL:

- a. Stations which are not on line (powered off), are deleted from the autopoll poll string.
- b. If a message cannot be sent (terminal in LOCAL or XMT mode), all terminals are autopollled once before going back and selecting again.
- c. The first time transmission is attempted, a FASTSELECT is used. If the FASTSELECT fails, a normal select is used to conserve line time.
- d. This CONTROL is designed to handle line groups and switched lines.
- e. If a station has a message and it is powered off or taken off line, no more attempts are made to send the message to the station until the station can be successfully polled again.

The AUTODYNCTL CONTROL does an automatic reconfiguration every three minutes. For larger configurations, the user should increase this interval to avoid continuous reconfigurations. Reconfiguration is accomplished by placing previously dead stations into the poll string. In order to avoid interfering with active stations, the dead stations are added to the end of the poll string. These stations are removed from the poll string if they time out again.

Once a station is active, it is not removed from the poll string unless it times out three consecutive times, or is made NOT READY.

The autopoll buffer declared in the LINE Section of the Network Controller must be equal to five times the number of stations on the line plus five.

If the LINE(QUEUED) bit is set and no stations are responding and the line was established as switched, it is disconnected; otherwise, all stations are again single-pollled to determine if any stations have come on line or changed status to allow LINE(QUEUED) to become false. Any exception on output is reported to the user program when RETRY is exhausted.

Refer to the comments at the beginning of POLTCTDDYN, SELTCTDDYN, and AUTODYNCTL in the NDL/LIBRARY for a full explanation of the purpose of each variable used by the AUTODYNCTL CONTROL.

The CANDETDCTL control is intended to service stations with terminals of type TD or TC series. The CANDETDCTL CONTROL and the REQUESTs perform the same as the standard CONTROL AUTOPOLCTL with the following exceptions.

- a. If 24 consecutive messages are sent to a station and more messages to that station are still queued, output activity is temporarily interrupted, and AUTOPOLL is initiated in order for input to be handled. CANDE queues only one full screen of messages to a terminal at a time. CANDE can be informed of a terminal's screen size in number of 80-character lines. Typically, only one message is sent in response to one input from the terminal.
- b. The autopoll poll string containing the station addresses is maintained dynamically as explained in the comments in the CANDETDCTL listing. Basically, only those stations that are on line (responding with an EOT or SOH when polled) are kept in the poll string. At a fixed time interval of 5 minutes, all stations previously considered as not on line are again included in the poll string in order to redetermine their on line or off line status. If none of the stations respond as being on line, each one is single-polled (rather than autopoll) in turn until at least one station is found to respond again.

AUTOPOLL is initiated with a TIMER value depending on the current condition of the line. This TIMER value is set as follows:

1. One second if output is queued for any station on the line; else
 2. Two minutes if input from any station is disabled; else
 3. Infinite, but interruptable.
- c. If no output is queued for a line, and all stations are NOT READY or input is disabled, each station is single-polled once the line is idled for two minutes after which the CONTROL is re-entered.

POINT-TO-POINT CONTROLS

This class of CONTROL handles a line with more than one station attached except in the case of the CONVERCTL CONTROL which may only have one station defined per line.

Contention Control

The basic purpose of the PTPTCONCTL CONTROL is to initiate the appropriate input or output REQUEST to a station on the line. Currently, there may be only one station assigned to a line using this CONTROL if REQUESTs TCTUPTXMIT, TCTUPTRCV, or TCTUPTIO are utilized. However, several stations may be concatenated on a single line if REQUESTs TDPTXMIT, TDPTRCV, or TDBATCHXMT are used.

Upon completion of a previous REQUEST when LINE(CONTROL KEY) equals 0, or notification of system reconfiguration when LINE(CONTROL KEY) equals 1, the CONTROL checks the station on the line. If the station is READY, QUEUED, and MYUSE is not INPUT only, the output REQUEST is initiated. If the station is READY, ENABLED, and MYUSE is not OUTPUT only, the input REQUEST is initiated. The failure to initiate either REQUEST for the station causes the line to be put in the idle state.

When the line has a READ with NO TIMEOUT pending and a message is queued for the line where there was no prior message in the queue, the CONTROL is entered with LINE(CONTROL KEY) equal to 2 and the CONTROL cancels the I/O in process. The CONTROL is then re-entered with LINE(CONTROL KEY) equal to 0.

If the station establishes itself on a switched line, the CONTROL is entered with LINE(CONTROL KEY) equal to 3 or LINE(CONTROL KEY) equal to 5 if the line is an Automatic Calling Unit (ACU), at which time the CONTROL sets LINE(TOG[5]) equal to 1. Upon failure to initiate any REQUEST for the station,

if LINE(TOG[5]) equals 1 and the line is queued, the line automatically disconnects; otherwise, the line is put into the idle state.

If a REQUEST performs a TERMINATE OUTPUT(RETURN) statement, the CONTROL is entered with LINE(CONTROL KEY) equal to 4. The CONTROL performs a CONTINUE statement which assigns an output message buffer to the line for the station, if one is queued, and returns to the REQUEST from which the TERMINATE OUTPUT(RETURN) statement was executed.

Conversational Control

The basic purpose of the CONVERCTL CONTROL is to initiate the appropriate input or output REQUEST to the station on the line. There may be only one station assigned to a line using this CONTROL since this is a point-to-point conversational line CONTROL.

Upon completion of a previous REQUEST, when LINE(CONTROL KEY) equals 0, or notification of system reconfiguration, when LINE(CONTROL) equals 1, this CONTROL checks the station on the line. If the station is READY, ENABLED, and MYUSE is INPUT/OUTPUT, the CONVERCTL CONTROL obtains an input/output buffer and initiates the REQUEST. If the station is queued but not enabled, the CONTROL initiates output only.

If the station establishes itself on a switched line, the CONTROL is entered with LINE(CONTROL KEY) equal to 3 or LINE(CONTROL KEY) equal to 5 if the line is an Automatic Calling Unit (ACU), at which time the CONTROL sets (TOG[5]) equal to 1. Upon failure to initiate any REQUEST for the station, if LINE(TOG[5]) is equal to 1 and the line is queued, the line automatically disconnects; otherwise, the line is put into the idle state.

If a REQUEST performs a TERMINATE OUTPUT(RETURN) statement, the CONTROL is entered with LINE(CONTROL KEY) equal to 4. The CONTROL performs a CONTINUE statement which assigns an output message buffer to the line for the station, if one is queued, and returns to the REQUEST from which the TERMINATE OUTPUT(RETURN) statement was executed.

Remote Job Entry (RJE) Control

RJECTL is a special RJE CONTROL that operates in a manner the same as the CONVERCTL CONTROL, except that it may be used to handle a line which has more than one station attached.

Four stations must be declared for each line declared in the Network Controller, and only one line group is permitted. LINE(TALLY[6]), LINE(TALLY[7]), and LINE(TALLY[8]) are used by this CONTROL to govern the stations on the line or the stations on each line within the line group.

Only the special input/output REQUESTs named RJE and RJEHOST are recommended for use with this CONTROL.

Refer to the comment section of the RJE and RJEHOST REQUESTS in the NDL/LIBRARY for a complete description of the purpose of each variable used by the RJECTL CONTROL.

ASSOCIATED REQUESTS AND CONTROLS

Many combinations of REQUEST and CONTROL sets can be used together. Refer to table E-1 for the associated REQUESTs and CONTROLs that can be used together.

Table E-1. Associated REQUESTs and CONTROLS

	CONTROL NUMBER						
	1	2	3	4	5	6	7
POLLTCTD	X	X					
POLTCTDDYN						X	
SELECTCTD	X	X					
SELTCTDDYN						X	
FASTSELTC	X	X					
TCTUPTXMIT			X				
TCTUPTRCV			X				
TCTUPTIO				X			
TDPTXMIT			X				
TDPTRCV			X				
TDBATCHXMT			X				
WRITETTY			X				
READTTY			X				
RJE					X		
RJEHOST					X		
CANDEPOLTD							X
CANDESELTD							X
CANDEFSLTD							X
CANDEIOTTY				X			
DIAGTCTDIO	X	X					

REQUEST NAME

X = recommended usage.

<u>Control Number</u>	<u>Control Name</u>
1	POLSELCTL
2	AUTOPOLCTL
3	PTPTCONCTL
4	CONVERCTL
5	RJECTL
6	AUTODYNCTL
7	CANDETDC

ASSOCIATED TERMINAL AND REQUEST SETS

Table E-2 lists the REQUEST sets that can be used for each terminal.

Table E-2. Associated Terminal and REQUEST Sets

R E Q U E S T N A M E	TERMINAL NUMBER								
	1	2	3	4	5	6	7	8	9
POLLCTD	X	X	X	X	X	X	X		
POLTCTDDYN	X	X	X	X	X	X	X		
SELECTCTD	X	X	X	X	X	X	X		
SELTCTDDYN	X	X	X	X	X	X	X		
FASTSELCTD	X	X	X	X	X	X	X		
TCTUPTXMIT	X		X						
TCTUPTRCV	X		X						
TCTUPTIO	X		X						
TDPTXMIT					X				
TDPTRCV					X				
TDBATCHXMT	X	X							
WRITETTY								X	
READTTY								X	
RJE									X
RJEHOST									X
CANDEPOLTD		X			X				
CANDESELTD		X			X				
CANDEFSLTD		X			X				
CANDEIOTTY								X	
DIAGTCTDIO	X	X	X	X	X	X	X		

X = Recommended usage.

<u>Terminal Number</u>	<u>Terminal Type</u>
1	TC500, TC700
2	TC3500, TC4000, TC5000
3	TU500
4	TU700
5	TD700, TD800, TD820, TD830
6	TT142, TT602
7	S1200, S1700
8	TELETYPE, REDACTOR I (TTY)
9	REMOTE APPLICATIONS, SYSTEM TO SYSTEM

INDEX

- < DIGIT >, 3-2, 3-3
- < LETTER >, 3-1
- < SLASH >, 3-2
- < SPACE >, 3-2
- < SPECIAL CHARACTER >, 3-2
- < VERTICAL BAR >, 3-2

- \$LIBRARY, E-1
- \$MERGE, 5-1
- \$NEW, 5-1

- ACCESSERR, 3-14, 3-15
- ACU, E-13
- ACU, adapter, 3-9
- ADDERR, 3-14, 3-40
- ADDRESS (RECEIVE), 3-40
- ADDRESS (TRANSMIT), 3-40
- ADDRESS SIZE Statement, 3-68
- ADDRESS Statement, 3-58
- ADDRESS STATION, 3-40
- ASCII, 3-35
- ASSIGNMENT Statement, 3-30, 3-49
- ATTACH, 4-5, 4-15, 4-22
- ATTACH OUTPUT Statement, 3-30
- ATTACH REPLY, 4-15
- ATTACH Statement, 3-30
- ATTACH-REPLY, 4-5
- ATTACHED, 4-22
- ATTACHOUTPUT, 3-40
- AUDIT, 3-8, 3-10, 3-31, 3-32, 6-1
- AUDIT Statement, 3-30
- AUDITFILE, 3-23, E-9
- AUDITFILE Declaration, 3-21
- AUTODYNCTL, E-3, E-6, E-14
- AUTOPOLCTL, E-13, E-15
- AUTOPOLL, 3-53, E-14, E-15
- AUTOPOLL SIZE, 3-55

- Backus-Naur Form (BNF), 3-1
- BNF, 3-1
- Bracket, 3-4
- BREAK, 3-14, 3-15
- BREAK I/O, 3-42
- BUFFER, 3-31
- BUFFERSIZE, 3-14, 3-40
- BUFFERSIZE Statement, 3-58

- CANDE, E-3, E-4, E-6, E-7, E-10, E-15
- CANDEFSLTD, E-6, E-7
- CANDEIOTTY, E-10
- CANDEPOLTD, E-3, E-4
- CANDESELTD, E-6
- CANDETDCTL, E-3, E-15
- CASE, 3-30, 3-32, 3-39, 3-48, 3-49, 3-54
- CASE Statement, 3-31, 3-32, 3-49

- CHANGE, 3-53, 3-59
- CHANGE Message, 4-30
- CHANGE REPLY Message, 4-30
- CHAR (CHARACTER), 3-7
- CHARACTER (CHAR), 3-7
- Character set, 3-1
- CHECK Option, 5-4
- CLOSE, 4-22
- CLOSE Message, 4-24
- COBOL, 2-3, 3-71
- CODE Option, 5-4
- CONSTANT, 3-41
- CONSTANT Declaration, 3-23, 3-24
- CONTINUE, 2-4, 3-9
- CONTINUE Statement, 3-49
- CONTROL, 2-2, 3-5, 3-9, 3-11
- Control cards, 5-3
- CONTROL KEY, 3-48
- CONTROL Option, 5-4
- CONTROLLER Statement, 3-62
- Controller Station, 3-12
- CONVERCTL, E-9, E-15, E-16
- CONVERT, 3-32
- CONVERT Statement, 3-17, 3-18
- CR, 3-35, 3-38
- CREATE LIBRARY Option, 5-4

- Data Message Format, 4-6
- DC/AUDIT, 6-1
- DCWRITE, E-9
- DECIMAL, 3-32
- DECIMAL Statement, 3-17
- DECLARATION, 2-3
- DECLARATION Section, 3-7, 3-21
- DECREMENT Statement, 3-33
- DEFAULT Statement, 2-6, 3-58
- DEFINE Statement, 3-25
- DETACH, 4-22
- DETACH Message, 4-23
- DETACH OUTPUT Statement, 3-33
- DETACH-REPLY, 4-2
- DETACH-REPLY Message, 4-23
- DIAGNOSTIC Statement, 3-59
- DIAGTCTDIO, E-9
- DISPLAY, 3-10, 3-34
- DISPLAY Statement, 3-33, 3-49, 3-50
- DO, 3-30, 3-39, 3-49, 3-54
- DO Statement, 3-34, 3-50, 3-51
- DOUBLE Option, 5-4
- DUMMY, 3-72, 4-1
- DUMP Statement, 3-35, 3-51
- DYNAMICSIZE Option, 5-4

- EBCDIC, 3-1, 3-2, 3-3, 3-17, 3-24, 3-35
- ENABLE, 3-42

INDEX (Cont)

- ENDOFBUFFER, 3-14, 3-15, 3-35, 3-40, 3-55
- EOT, 3-35, 3-38
- Equal (EQ), 3-4
- Error Flags, 3-14
- EXCEPTION, 3-15
- EXPRESSIONS, 3-19

- FAMILY Statement, 3-71
- FASTSELECT, E-6, E-7, E-14
- FETCH, 3-7
- FETCH Statement, 3-35
- FILE, 2-1, 2-3, 2-4
- FILE DEFAULT Statement, 3-72
- FILE Section, 3-71
- FILE Statement, 3-23, 3-71
- Files, 2-6
- FINISH Statement, 3-35
- FINISH TRANSMIT-INITIATE RECEIVE
(NO TIMEOUT), 3-53
- FLAGS, 3-32
- FORGETERRORS Option, 5-4
- FORMATERR, 3-15, 3-41
- FREQUENCY, 3-48
- FREQUENCY (INPUT), 3-7
- FREQUENCY (OUTPUT), 3-7
- FREQUENCY Statement, 3-63

- Glossary Of Terms, B-1
- GOOD RESULTS, 4-5
- GOOD RESULTS REPLY, 4-5
- GOODRESULTS REPLY, 4-7
- Greater than (GT), 3-4
- Greater than or equal (GE), 3-4

- HEADER, 4-1

- Identifiers, 3-2
- IF, 3-39, 3-49, 3-54
- IF Statement, 3-35, 3-51
- INCREMENT Statement, 3-37
- INITIALIZE RETRY, 3-38, 3-65
- INITIALIZE Statement, 3-37
- INITIALIZE TALLY, 3-15, 3-38
- INITIALIZE TEXT, 3-38
- INITIALIZE TOGGLE, 3-15, 3-38
- INITIALIZE TRAN (RECEIVE), 3-38
- INITIALIZE TRAN (TRANSMIT), 3-38
- INITIATE AUTOPOLL, 2-4, 3-53
- INITIATE CANCEL, 3-9, 3-53
- INITIATE IDLE, 3-8, 3-53
- INITIATE INPUT, 3-51, 3-52, 3-53
- INITIATE INPUT (NOBUFFER), 3-52
- INITIATE INPUTOUTPUT, 3-53
- INITIATE OUTPUT, 3-52, 3-53
- INITIATE OUTPUT (NOBUFFER), 3-53
- INITIATE RECEIVE, 3-38, 3-40
- INITIATE RECEIVE (NO-TIMEOUT), 3-53
- INITIATE Statement, 3-38, 3-52
- INITIATE TRANSMIT, 3-38
- INPUT LOGICALACK, 4-5
- INPUTATTACHED, 3-8, 3-52
- Integers, 3-3
- IODESC, 3-8
- IOLOG, 6-1

- LENGTH (INPUT), 3-8
- LENGTH (OUTPUT), 3-8
- Less than (LT), 3-4
- Less than or equal (LE), 3-4
- LIBINFO, 5-4
- LIBRARY Option Statement, 5-4
- LINE, 2-1, 2-4, 3-8, 3-43, 3-55
- LINE (CONTROL KEY), 3-8
- LINE (QUEUED), 3-10
- LINE AUTOPOLL Statement, 3-68
- LINE CONTROL, 2-3, 2-4, 3-43
- LINE CONTROL Statement, 3-69
- LINE DEFAULT Statement, 3-69
- Line Section Statement, 3-67
- LINE TALLY, 3-16
- LINE TOG, 3-15
- LINE TYPE Statement, 3-70
- Lines, 2-6
- List of Required Statements, C-1
- LIST Option Statement, 5-4
- LOGIACKLACK REPLY, 4-5
- Logical Station Number, 3-11
- Logical Values, 3-4
- LOGICALACK Statement, 3-64
- LOSS OF CARRIER, 3-15
- LOSSOFDSR, 3-15

- Master Station, 3-12
- MAXBUFFERS Declaration Statement, 3-27
- MAX FILES Declaration Statement, 3-27
- MAX MESSAGES Declaration Statement, 3-27
- MAX TALLY, 3-16, 3-26
- MAX TALLY Declaration Statement, 3-26
- MAX TOG, 3-16, 3-26
- MAX TOG Declaration Statement, 3-26
- MAXSTATIONS, 3-10
- MCP, 2-2, 2-3, 3-7, 3-23, 4-1
- MCS, 2-1, 2-2, 2-3, 3-11, 3-33, 3-38, 3-42, 3-43,
3-53, 3-59, 3-65, 3-72, 4-1, 4-2
- MCS (Message Control System), 4-1
- MCS Messages and Replies, 4-2
- MERGE Option, 5-4
- Message Control System, 2-1, 2-3, 3-13

INDEX (Cont)

- MLC, 3-14
- MYUSE, 3-5, 3-55, 3-64
- MYUSE Statement, 3-64

- NC, 2-2
- NDL, 1-1, 2-1, 2-2, 2-3, 2-4, 3-1, 3-2, 3-3, 3-5, 3-7, 3-10, 3-11, 3-57
- NDL Compiler, 1-1, 2-5, 2-6, 5-1
- NDL Compiler, 3-49
- NDL Error Handling, 7-1
- NDL LIBRARY, E-1
- NDL Sample Program, D-1
- NDL/DUMP, 3-35, 3-51
- NDL/DUMP Program, 7-2
- NDL/LIBRARY, E-14
- Network Controller, 2-1, 2-2, 2-3, 3-13
- Network Controller Diagnostic Aids, 6-1
- Network Definition Language, 1-1, 2-1, 2-3, 3-1
- Network Information File, 2-1, 2-4
- NEW Option, 5-4
- NEWSOURCE, 5-1
- NIF, 2-1, 2-3, 2-4, 3-26, 3-28, 5-1
- NIF Declaration Statement, 3-26
- NO TIMEOUT, 3-9
- Normal Station, 3-12
- Not Equal (NE), 3-4
- NSSIZE Option, 5-4
- NULL, 3-49
- NULL Statement, 3-39, 3-54

- OPEN, 4-5, 4-9, 4-22
- OPEN REPLY, 4-9
- Operators, 3-3
- OUTPUTATTACHED, 3-8, 3-43, 3-53

- PAGE Option, 5-4
- PARITY, 3-15, 3-38
- PARTICIPATING, 4-5
- POLL, 2-4, 3-53
- POLL Statement, 3-54
- POLLCTDDYN, E-3
- POLLTCTD, E-1, E-2, E-3, E-4, E-9
- POLSELCTL, E-1, E-12
- POLTCTDDYN, E-6, E-14
- Primary, 4-5
- Primary files, 4-2
- PTPTCONCTL, E-15

- QUEUE INPUT Statement, 3-39
- QUEUE OUTPUT, 3-30
- QUEUE OUTPUT Statement, 3-39

- READ, 3-8, 3-9
- READTTY, E-5
- RECALL Message, 4-31
- RECALL-REPLY Message, 4-31
- RECALL-REPLY, 4-5
- RECALLED, 4-5
- RECEIVE, 3-35
- RECEIVE Statement, 3-40
- RECEIVE TEXT, 3-42
- Related Document, 1-1
- Relational Operators, 3-3, 3-4
- Relative Station Number, 3-11
- REMOTE FILE INFO Message, 4-32
- REMOTE FILE INFO REPLY Message, 4-32
- Remote File Interface, 4-1
- REMOTE FILE OPEN OR CLOSE, 3-8
- Remote Job Entry, 3-9
- REMOTE-FILE-NO, 4-5
- REMOVE Message, 4-31
- REMOVE-REPLY Message, 4-31
- REQUEST, 2-2, 2-3, 3-5, 3-9, 3-43, 3-60
- REQUEST Statement, 3-59
- Reserved Words, A-1
- RESIDENT Statement, 3-72
- RESULTDESC, 3-10
- RETRY, 3-10
- RETRY Statement, 3-65
- Ring complete, 3-9
- RJE, 3-9, E-10, E-11, E-16
- RJE/HOST, E-11, E-12
- RJECTL, E-16
- RJEHOST, E-16

- S-code, 3-1
- Secondary, 4-5
- Secondary files, 4-2
- SELECTCTD, E-1, E-5, E-6
- SELECTDDYN, E-14
- SELECTTCID, E-9
- SELTCTDDYN, E-6
- Separators, 3-5
- SEQ (SEQUENCE) Option, 5-5
- SEQUENCE, 3-10
- SGL (SINGLE) Option, 5-5
- Sharing Recourse Among Multiple MCS Files, 4-2
- Slave Station, 3-12
- SLC, 3-14
- Source disk, 5-3
- Source records, 5-2
- STATION, 2-1, 2-3, 2-4, 3-7, 3-9, 3-10, 3-11, 3-64
- STATION (ENABLED), 3-11
- STATION (LINE), 3-11
- STATION (MYUSE), 3-11, 3-52, 3-53
- STATION (QUEUED), 3-11
- STATION (READY), 3-11
- STATION (TYPE), 3-12
- STATION (VALID), 3-13
- STATION ADDRESS Statement, 3-65

INDEX (Cont)

- STATION DEFAULT, 3-42
- STATION DEFAULT Statement, 3-62, 3-65, 3-66
- STATION QUEUED, 3-48
- STATION READY Statement, 3-67
- STATION TYPE, 3-55
- Stations, 2-6
- STATUS Message, 4-25, 4-26
- Status Reply Message, 4-26
- STRING, 3-41
- STRING Statement, 3-17
- Strings, 3-3
- SUPPRESS Option, 5-5
- System Status Variables, 3-6

- TALLIES, 3-32
- TALLY, 3-4, 3-38, 3-40, 3-54
- TALLYs and TOGs, 3-15
- TCTUPTIO, E-9, E-15
- TCTUPTRCV, E-4
- TCTUPTXMIT, E-7, E-15
- TDBATCHXMT, E-8, E-15
- TDPTRCV, E-4, E-15
- TDPTXMIT, E-7, E-15
- Telephone Number, 3-9
- Teletype, 3-38
- TERMINAL, 2-1, 2-3, 2-4, 3-29
- TERMINAL DEFAULTS Statement, 3-59
- TERMINAL TYPE, 3-13
- Terminals, 2-6
- TERMINATE, 3-11, 3-35
- TERMINATE DISCONNECT, 3-42
- TERMINATE ERROR, 3-15, 3-16, 3-42, 4-7
- TERMINATE INPUT, 3-15, 3-16, 3-42
- TERMINATE INPUT (RETURN), 3-42, 3-52
- TERMINATE INPUT (RETURN, NOBUFFER), 3-42
- TERMINATE LOGICALACK, 3-42, 4-5
- TERMINATE NOINPUT, 3-43
- TERMINATE OUTPUT, 3-43
- TERMINATE OUTPUT (RETURN), 3-9, 3-43, 3-53
- TERMINATE RELEASE, 3-68
- TERMINATE RELEASE (STATION), 3-43
- TERMINATE Statement, 3-41
- TEXT, 3-30, 3-32
- TIME, 3-7, 3-13, 3-16
- TIME (TALLY), 3-13
- TIMEOUT, 3-8, 3-15, 3-38
- TOG, 3-4, 3-15, 3-40
- TOGGLES, 3-32
- TRAN (RECEIVE), 3-13, 3-41
- TRAN (TRANSMIT), 3-13, 3-41
- TRANERR, 3-13, 3-15
- TRANSLATE, 3-35, 3-38
- TRANSMISSION NUMBERS, 3-37
- TRANSMISSION Statement, 3-61
- TRANSMIT, 3-45
- TRANSMIT ADDRESS, 3-44
- TRANSMIT CHARACTER, 3-44
- TRANSMIT Statement, 3-43
- TRANSMIT TALLY, 3-44
- TRANSMIT TOG, 3-44
- TRANSMIT TRAN (RECEIVE), 3-44
- TRANSMIT TRAN (TRANSMIT), 3-44
- TYPE Statement, 3-61

- UNDO, 2-4, 3-30, 3-32, 3-49, 3-51
- UNDO Statement, 3-56
- UNPROCESSED OUTPUT, 4-5
- UPDATE LIBRARY, 5-5
- UPL, 2-3
- USER Message, 4-34

- VA OF ID, 3-71
- Variables, 3-5
- VOID Option, 5-5
- VSSIZE Option, 5-5

- WRITE/READ, 3-9
- WRITETTY, E-8

- ZIP, 4-1