60480500

CONTROL DATA CORPORATION

NETWORK PRODUCTS

NETWORK PRODUCTS STIMULATOR VERSION 1 REFERENCE MANUAL

CDC® OPERATING SYSTEM:
NOS 1

REVISION RECORD	
REVISION	**DESCRIPTION**
A	Original release at PSR level 472.
(03-31-78)	
B	Revised at PSR level 477 for technical corrections.
(08-15-78)	
C	Revised at PSR level 485 for technical corrections.
(12-18-78)	
D	Revised to reflect the release of NPS 1.1 which includes support of input/output line speed delay,
(08-10-79)	abbreviated input statement keyword, execution from system origin and nested repeats.
Publication No. 60480500	

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Publications and Graphics Division
215 MOFFETT PARK DRIVE
SUNNYVALE, CALIFORNIA 94086

or use Comment Sheet in the back of this manual

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

Page	Revision
Cover	-
Title Page	-
ii	D
iii/iv	D
v/vi	D
vii	D
viii	D
ix	D
1-1	D
1-2	D
2-1 thru 2-10	D
3-1 thru 3-7	D
4-1	A
4-2 thru 4-6	D
5-1	B
5-2 thru 5-5	D
5-6	A
A-1	C
A-2	A
A-3	D
A-4	D
A-5	A
B-1 thru B-10	D
C-1 thru C-3	A
D-1	D
Index-1	D
Index-2	D
Comment Sheet	D
Mailer	–
Back Cover	–

Page	Revision

Page	Revision

PREFACE

This manual describes the CONTROL DATA® Network Products Stimulator Version 1. The Network Products Stimulator provides a method of testing and exercising network host products software; it is not an interactive test tool but rather a tool to test interactive systems.

Through use of the Network Products Stimulator, the Network Access Method and applications using it can be dynamically exercised as though by a live terminal network. Controlled terminal sessions are input to the Network Access Method from a previously prepared library of user-written scripts that simulate actual terminal sessions.

As described in this publication, Network Products Stimulator Version 1 operates under control of the NOS 1 operating system for the CDC® CYBER 170 Series; CYBER 70 Models 71, 72, 73, and 74; and 6000 Series Computer Systems.

The reader is assumed to be familiar with the network products and the NOS 1 operating system as described in the publications listed below.

The following CDC publications provide additional information pertinent to the use of NPS.

Publication	Publication Number
Network Products Network Access Method Version 1 Reference Manual	60499500
Network Products Network Access Method Version 1 Network Definition Language Reference Manual	60480000
Network Products Remote Batch Facility Version 1 Reference Manual	60499600
Network Products Interactive Facility Version 1 Reference Manual	60455250
Network Products TAF Version 1 Reference Manual	60455340
NOS Version 1 Operator's Guide	60435600
NOS Version 1 Reference Manual (Volume 1)	60435400

CDC manuals can be ordered from Control Data Literature and Distribution Services, 308 North Dale Street, St. Paul, Minnesota 55103.

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

CONTENTS

APPENDIXES

INDEX

FIGURES

NOTATIONS USED IN THIS MANUAL

Throughout this publication, the following conventions are used in the presentation of statement formats. Additional notations applicable only to the =SCRIPT directives appear in section 2.

UPPERCASE	letters indicate words, acronyms, or mnemonics either required by the network software as input to it, or produced as output.
lowercase	letters represent symbols supplied by the Network Access Method (NAM), by the terminal user, or by the network software as output. When generic terms are repeated in a format, a subscript is appended to the term for identification.
. . . Ellipsis	indicate that omitted entities repeat in the form and function of the entity last given.
[] Brackets	enclose optional entities.
{} Braces	enclose entities from which one must be chosen.

Unless otherwise specified, all numbers are decimal values.

NETWORK PRODUCTS STIMULATOR 1

Network Products Stimulator (NPS) is an evaluation package used to test the Network Access Method (NAM) and the applications using NAM. NPS allows a controlled message load to be presented to the NAM software without the use of external communications equipment or related terminals. Because NPS runs as a batch job with no exceptional system requirements, it provides a convenient method of exercising and testing network applications with minimal disruption of normal system operations. NPS can be run as the only front-end in the system, or in conjunction with a live network. Both batch and interactive terminals can be simulated.

NPS simulates a terminal network through user-created scripts that describe various terminal sessions. The number of terminal sessions that can be simulated simultaneously is limited only by the size of central memory and network host products limits. More central memory is required by NPS as the number of terminals increases. The terminal sessions appropriate for a particular test must be created in advance and supplied to NPS when the stimulation is executed.

During NPS execution, a controlled message load is transmitted to the appropriate application through the NAM interface. Application response to each message is the same as if an actual terminal had supplied the message. NPS records the stimulation session as it occurs and can prepare a report for use in analyzing performance of an application.

NPS is composed of three separate utilities, SCRIPT, STIM, and REPORTR, which can be executed at different times:

- SCRIPT creates a script library of terminal sessions to be simulated. (A user-written description of a single terminal session is called a script; a collection of scripts in the format used for stimulation is called a script library.) SCRIPT requires 35000_8 central memory words for execution and 65000_8 central memory words for compilation.

- STIM performs the stimulation. It uses the script library created by SCRIPT, the local configuration file (LCF) used by the Communications Supervisor, the network configuration file (NCF) used by the Network Supervisor, and the STIM input statements that describe the terminals to be simulated. During execution, STIM writes a log file of messages transmitted to the applications, and records the replies received from the applications. STIM requires at least 40000_8 central memory words for execution and 70000_8 central memory words for compilation. STIM also requires a dedicated peripheral processor, and thus cannot be swapped out.

- REPORTR extracts data from the log file created by STIM, performs a statistical analysis on the data, and creates a specified report. REPORTR requires 50000_8 central memory words for execution and 70000_8 central memory words for compilation.

The three stages of NPS, the functions of each stage, and the interface between STIM and NAM are illustrated in figure 1-1.

Part A of figure 1-1 illustrates the operation of SCRIPT. SCRIPT creates a new script library as specified by the SCRIPT directives. The directives can be maintained on an UPDATE OLDPL or a MODIFY OPL and modified as desired before submission to the SCRIPT compiler. The SCRIPT directives describe a terminal session, specifying information such as the messages to be sent, directions on when to send them, acceptable responses, and acceptable times between responses. SCRIPT parameters also provide instructions for scripts and messages to be listed during creation of the new script library.

SCRIPT creates two types of output:

- A new script library containing scripts to be used in the stimulation. The script library file is written (using CIO format) in two parts. The first part contains all nonembedded data messages, in the format that STIM processes. The second consists of the compiled object code for all scripts.

- A listing containing scripts, errors and/or messages, as specified in the SCRIPT call.

Part B of figure 1-1 illustrates the processing that occurs during the stimulation. The central processor routine, STIM, is called by control statement, reads the input, and performs the stimulation generating dayfile messages and the log file as output. STIM reads two sources of input: the script library and a set of STIM input statements. The script library contains all scripts and messages describing the terminal sessions that have been created. The STIM input statements define the terminals that are to be simulated during this run and designate which script is to be run on which terminal.

The simulated network is determined by parameters on the STIM control statement and input statements. A STIM input statement designates each Network Processing Unit (NPU) to be simulated. A separate statement designates each terminal to be simulated. The NPUs and terminals named must have been defined in a Network Definition Language job which creates the local configuration file (LCF) that will be used by the network software.

A STIM run is initiated by a batch or system origin job containing the STIM call statement and the necessary STIM input statements. STIM then executes as a normal batch origin type user job, when it is initiated by a batch job. STIM runs as a subsystem for a system origin job.

Once a STIM run is started, all simulated terminals will login, perform the script tests, and logout as specified by the scripts in use.

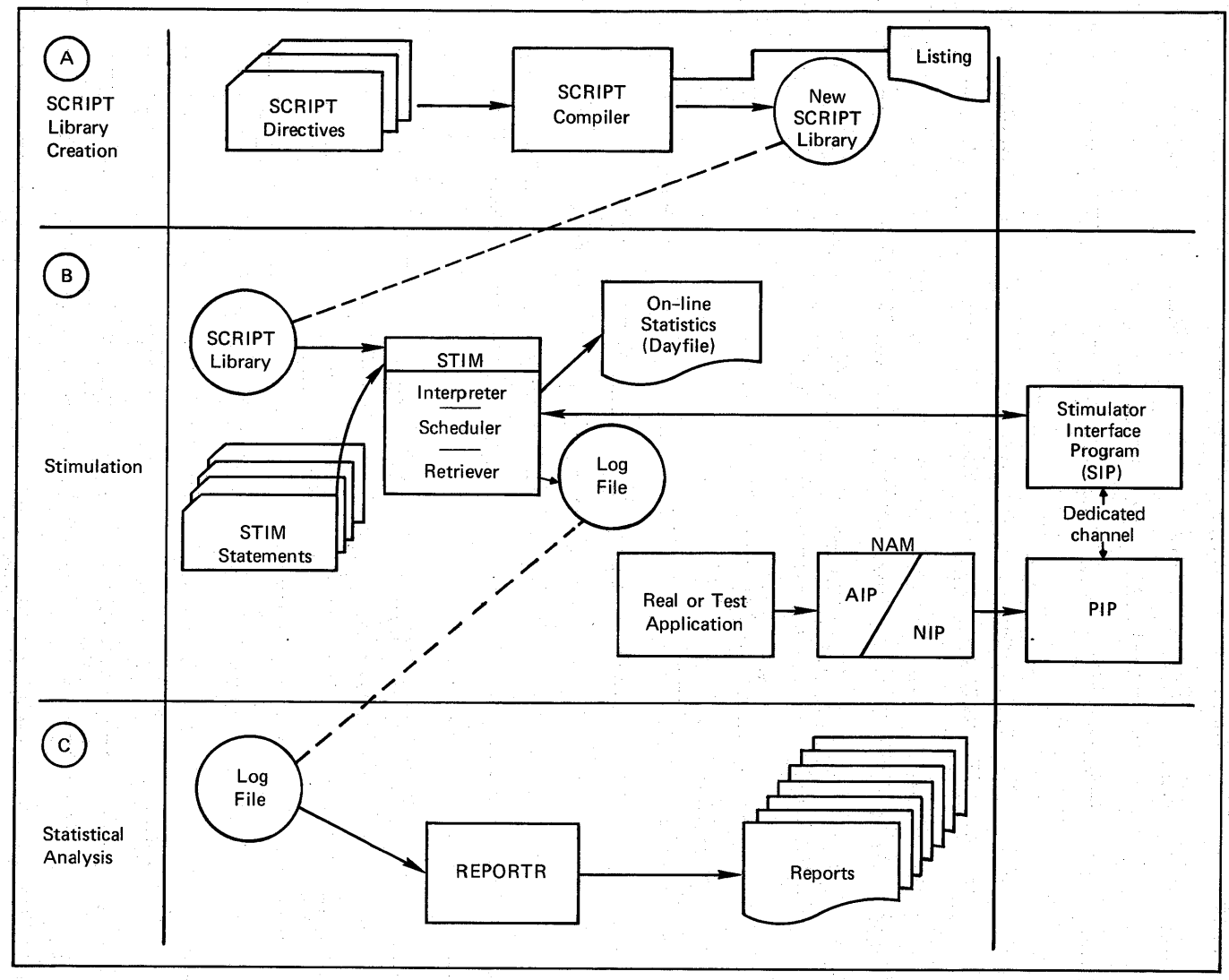

Figure 1-1. Network Products Stimulator Environment

During the stimulation, STIM supplies SIP, the Stimulator Interface Program, with messages to be sent from the simulated terminals as directed by the scripts. SIP, running in a dedicated peripheral processor, sends the messages over a dedicated data channel to the NAM Peripheral Interface Program, PIP, which in turn sends the messages on to NAM and the applications using NAM. SIP communicates with PIP in the same manner as the NPU does, sending and receiving messages across the dedicated data channel and emulating the hardware and software control functions of the NPU. The responses come back to SIP and STIM through the reverse procedure. STIM records the messages and responses on a log file.

Part C of figure 1-1 illustrates the processing that occurs when analyzing the data. REPORTR reads the log file and produces a report containing two types of data:

- A dump of the messages sent and received during a specified time slice.
- A statistical analysis of the messages sent and received during another specified time slice.

Parameters on the REPORTR control statement specify the time slices reported.

An NPS run normally consists of four steps:

1. The first step is a manual operation. The user must write a description of the terminal workload using SCRIPT directives. The complete terminal workload is called a script.

2. The second step is a batch job (SCRIPT), which compiles and translates the user-written scripts into object time scripts. These object time scripts are written to the Script Library for later use by STIM.

3. The third step is the actual stimulation/simulation run executed by STIM. STIM configures the simulation environment according to parameters specified, and maps the scripts from the Script Library to the terminals configured. It then performs the test by following the compiled script procedures from the Script Library.

4. The final step is the execution of the batch job REPORTR that analyzes the data on the log file and generates selected reports.

A script is a user-oriented description of a predefined, semicontrolled dialog between a terminal operator and a network application. This dialog consists of user-written directives used to define terminal sessions.

LANGUAGE CONCEPTS

The Script Definition Language, an integral part of the Network Products Stimulator package, consists of a number of directives the script writer can use to define terminal sessions. These directives can be maintained on an Update or Modify program library for subsequent use with the SCRIPT compiler.

The Script Definition Language consists of three types of directives:

- Pseudo directives
- Declarative directives
- Procedural directives

Pseudo directives establish basic script characteristics; for example, they designate script beginning and end. Pseudo directives are processed by the SCRIPT compiler, but are not executed by STIM.

Declarative and procedural directives both direct STIM to perform routine simulation. The difference between them is based on when they are executed. Declarative directives declare and define test conditions that are to be acted upon at some later time during execution of the script, whereas procedural directives invoke a procedure that is immediately acted upon during execution of the script.

Script Definition Language directives allow the script writer to:

- Define and save all messages to be sent from the terminal to the host application software.
- Save output responses to specific input for purposes of response verification.
- Include timing dependencies relative to the beginning of the simulation or delay time between messages.
- Do multiple value condition tests and vary the execution of the script accordingly.

Script directive syntax errors or parameter errors are detected by the SCRIPT compiler. They cause termination of the compiler run with a diagnostic message. The SCRIPT compiler scans the directive until the statements format is satisfied. Any additional information, beyond the basic format, is normally ignored by the compiler.

The nominal unit of a script for a terminal session is a message. The script causes a simulated terminal operator message to be sent to an application, which processes the message and returns a response.

NOTATION

The following notation is used throughout the description of the Script Definition Language and supplements the conventions given in the Preface:

= -- All Script Definition Language directives begin with an = sign in column 1 followed by the directive starting in column 2.

=X -- An X character immediately after the = sign denotes that a required prefix character (B, I, or C) must be substituted for the X. The B character indicates the directive is of batch mode, and the I character indicates the directive is of interactive mode. The character C is used only with messages intended for conditional IF MATCH directives.

blank or comma -- One or more blank characters or commas must separate all language keywords and parameters. A blank or comma is the only punctuation used. Extra commas or blanks are not required if an optional parameter is omitted. Periods are not allowed.

character -- Unless otherwise stated, the use of the term alphanumeric character or character implies any of the characters in the CDC Graphic Display Code 64-Character Set.

message -- Logical units of information transmitted between the host system and a terminal. Depending on the context, it can be a line of data, a file of information, or a screen of information. If long, it can be subdivided into blocks; that is, BLK,BLK,. . ., MSG. As defined in the Network Access Method Reference Manual, a BLK block contains a segment, but not the last segment, of a data message. A MSG block contains the last segment or all of a data message.

lfn -- Logical file name of one to seven letters or digits.

CHARACTER SET

All upline and downline nontransparent message data in the network is transmitted in ASCII. NPS makes use of the NOS extended 128-character set to represent message traffic data. That is, all message data is represented using the CDC 64-character set; but in order to represent and keypunch NPS-declared messages using the full 128-ASCII character set, two of the CDC graphic characters are designated as escape characters. The CDC graphic characters ≤ and ¬ (internal 6-bit codes 74 and 76) are the two escape characters. Thus, two CDC graphic characters are required to represent the two escape characters and all those characters beyond the first 64.

The NPS character set is given in appendix A. This is the character set that must be used in preparing message data for use by NPS.

In addition to the escape character designation to represent the full 128-ASCII character set, the following special characters are used.

End of Logical Line Character

NPS makes use of a special character in the NOS extended 128-character set to define a logical line delimiter. The special character ≤H (internal display code 7410_8) denotes the end of a logical line (carriage return).

Inclusion of this character in the message text results in the special character (≤H) and any characters which follow on that 80-character statement being discarded. The message text up to the special character is sent to the host as a MSG block containing a single logical line. (Refer to the Network Access Method Reference Manual for message characteristics.)

End of Physical Line Character

NPS makes use of a special character in the NOS extended 128-character set to define a physical line delimiter. The special character ≤I (internal display code 7411_8) denotes the end of a physical line (line feed).

Inclusion of this character in the message text results in the special character (≤I) and any characters which follow on that 80-character statement being discarded. The message text up to the special character is sent to the host as a BLK block containing a single physical line. (Refer to the Network Access Method Reference Manual for message characteristics.)

Cancel Character

Cancel processing is performed by NPS. NPS makes use of a special character in the NOS extended 128-character set to detect and perform cancel processing. The special character ≤J (internal display code 7412_8) is used to invoke cancel processing. Inclusion of this character in the message text results in a cancel MSG block being sent to the host. (Refer to the Network Access Method Reference Manual for characteristics of the cancel process.)

Transparent Delimiter Character

Transparent mode message text processing is supported by NPS. The NPS-defined NOS extended 128-character set works equally well to construct either character or transparent mode messages. The only requirement for transparent mode messages is that the transparent delimiter be included as the last character of the message text. This character is not included as input in the MSG block that is sent to the host.

PSEUDO DIRECTIVES

The script writer defines the bounds of a script by specifying a SCRIPT directive and an ENDSCRIPT directive. The script itself consists of those directives between SCRIPT and ENDSCRIPT. The script is written to the new script library with the name specified on the SCRIPT call statement.

The SCRIPT directive marks the beginning of the script and gives the script a name. The format is shown in figure 2-1.

=SCRIPT sss	{ASYNC | HASP | MODE4}
sss	The script name, one to three letters or digits.

Figure 2-1. SCRIPT Directive Format

The three types of terminals that can be specified in the SCRIPT directive correspond to the three types of terminals supported by this release. Each type is described fully in the Network Access Method Reference Manual. As these three types correspond to the three forms of the Terminal Interface Program (TIP), each is called a TIP-type. One of the TIP types must be specified for each script. Once designated, a script can only be used to simulate that type of terminal.

The ENDSCRIPT directive marks the end of the script defined in the SCRIPT directive. Script execution stops upon encountering the ENDSCRIPT directive, and the associated terminal becomes idle. The format of ENDSCRIPT is shown in figure 2-2.

=ENDSCRIPT

Figure 2-2. ENDSCRIPT Directive Format

MESSAGE DEFINITION

Messages used in the simulation can be defined in the script by three different methods. Each method results in a different scheme for storing messages and a corresponding access method for their retrieval. The three methods of defining messages available to the script writer are explained under the following three headings.

Random Access Messages

One method of defining messages allows the script writer to define a message with the XMESSAGE directive. The format is shown in figure 2-3.

=XMESSAGE mmmmmmm	
mmmmmmm	The unique message identifier, one to seven characters.
=BMESSAGE	Defines a batch message.
=IMESSAGE	Defines an interactive message.
=CMESSAGE	Defines a message that is to be used in the conditional IF MATCH directive.

Figure 2-3. XMESSAGE Directive Format

The message text begins in column 1 of the card immediately following the XMESSAGE directive. All text, including blanks, up to the next valid Script Definition Language directive is taken to be the message text. All messages are character data only as defined in appendix A. Each data statement can contain up to 80 characters of data. Messages defined with the =CMESSAGE directive will only be checked through 72 characters of data. If Update is used, the Update identifier will use columns 74 to 80. Messages defined in this manner need not be defined in the script in which they are referenced, but must occur within a script of the same TIP-type as the script in which they are referenced.

The message name mmmmmmm must be unique within IMESSAGE, BMESSAGE, or CMESSAGE directives of a given script TIP-type; for example, a message that is defined by any message directive in a HASP script (such as in =SCRIPT sss,HASP) is thereafter designated for HASP terminal use only. If a name is used to define an IMESSAGE in a script of TIP-type ASYNC, that name cannot be used elsewhere in the script library input to define another IMESSAGE in an ASYNC script; however, it can be used to define a CMESSAGE in an ASYNC script. The same name could also be used to define an IMESSAGE in a MODE4 script. It is possible to use the same message name to define as many as eight messages because there are three message usage types and three TIP-types (a BMESSAGE is not allowed in an ASYNC script).

All messages defined by XMESSAGE in the script input are accumulated by the SCRIPT compiler and included in the first part of the script library file. The script library file must reside on disk during a STIM run, as the messages are randomly accessed by STIM.

All Script Definition Language directives that access messages defined as random access file messages must include the message name mmmmmmm. Although messages of any length can be defined and accessed by this method, it is recommended that only long messages be defined in this manner, and that very short messages be embedded in the script.

Embedded Messages

This method of message definition and identification does not make use of any of the MESSAGE directives. Instead, the message is embedded in the script immediately following a script directive, which would otherwise reference a message named mmmmmmm. The message text begins in column 1 of the script input line immediately following the directive. All text, including blanks, up to the next valid Script Definition Language directive is taken to be part of the message.

Further discussion of embedded script messages is included in the discussions of those script directives that allow this method of message definition (see Data Message Input). Messages defined by this method can be used for upline transmission to the host system or for downline message verification.

Because this method embeds a message in the particular script that uses it, multiple copies of such a message can exist in the script library file. For this reason, it is recommended that only very short messages, such as terminal commands and application commands, be defined by this method. Longer messages (for example, job decks) should be defined by one of the MESSAGE directives or by the user-defined external sequential message file.

External Messages

This method of message definition does not use any of the MESSAGE directives. It employs an external file with a message that is prepared for upline transmission to the host system. The file is created by the user and accessed by use of the =XSEND FROM lfn directive (described later under Data Message Input). The file must exist as a local file during script compilation.

System End-of-Record and End-of-Information

Because the messages sent from a terminal, particularly a batch terminal, often contain system-defined end-of-record (EOR) and end-of-information (EOI) cards interspersed in the message deck, the following means are provided for the user to identify where they are to occur, without perturbing the SCRIPT compiler input stream:

=EOR (nn)	Indicates the position in the input deck of an end-of-record card. The nn is the logical record level number. If omitted, nn=0 is assumed.
=EOI	Indicates the position in the input deck of an end-of-information card.

LABEL DIRECTIVE

Using the LABEL directive, the script writer can define a label (a location in the script to which execution can transfer from another location in the same script). The label declared can be referenced in REPEAT, GO TO, all IF and most global directives. When control is transferred to the specified label, execution begins at the first executable directive following the LABEL directive. The format of the LABEL directive is shown in figure 2-4.

=LABEL eeeeeee

eeeeeee	The label name, one to seven letters or digits, that can be referenced in other directives in this script. Labels are local to the script in which they are defined; one script cannot jump to another script.

Figure 2-4. LABEL Directive Format

COMMENT DIRECTIVE

The script writer can include comments to explain the script by using the COMMENT directive. The comments appear in the script listing, but have no effect on the script execution: comments are not sent to the dayfile or the log file. SCRIPT processes the COMMENT directives and includes them in the listing, but does not save them on the script library or pass them on to STIM. The COMMENT directive, or its alternate form =*, has the format shown in figure 2-5.

```
{=*       }
{=COMMENT } comnt
```

comnt — The comment, which is any character data starting in column 4 or 10, through column 80 of the card image. A blank or comma must separate the directive from the comment. Because each comment is only a single card image, lengthy comments must be specified by a series of comment directives. Comments may appear after the =ENDSCRIPT directive.

Figure 2-5. COMMENT Directive Format

LIST DIRECTIVE

Listings of scripts can be obtained during a SCRIPT compiler run. The LIST directive should not be placed within the script (between SCRIPT and ENDSCRIPT). If LIST occurs within a script, a nonfatal error occurs and the directive is processed. The LIST directive is only in effect if the LO=Q or the LO=M option is selected on the SCRIPT call statement (described in section 3). The format of LIST is shown in figure 2-6.

```
=LIST  sss
```

sss — The name of the script to be listed. If script sss is not in the input stream, a nonfatal error occurs.

Figure 2-6. LIST Directive Format

DECLARATIVE DIRECTIVES

The declarative directives described here declare a process that is nonimmediate; that is, the process is declared but is not executed until necessary. These directives are continuously in effect in a particular script until they are respecified or cancelled in the same script. Because these directives are continuously in effect, they are called global directives. A global directive executed in one script has no effect on any other script.

Because it is not always possible to predict the order or content of response messages, the script writer needs a facility for checking responses to determine the next desired terminal action. Because execution of the script is sequential, it would prove cumbersome to include the desired test at each sequential step to test for the anticipated response.

Global directives permit out-of-line testing of many responses received from the application and need be specified only once. Global directives can be used in combination by the script writer to create a procedure, analogous to a subroutine, that is executed on demand.

LOG FILE CONTROL

The Stimulator normally records messages sent, responses received, and certain statistical and diagnostic messages on the log file in their entirety. Depending on the nature and state of checkout of a particular application, it might be desirable to record only part of the upline and downline message traffic on the log file for purposes of efficiency. The LOGGING IS directive, shown in figure 2-7, allows the script writer to log or not log message traffic as desired. OFF inhibits logging for that script until an ON directive is encountered, which remains in effect until OFF is encountered, etc. The initial state is ON.

```
=LOGGING IS {ON  }
            {OFF }
```

ON — Indicates normal mode and the initial state.

OFF — Inhibits message traffic from being logged or included in statistical reports.

Figure 2-7. LOGGING IS Directive Format

MESSAGE TIMEOUT CONTROL

When a message has been sent from a terminal to an application, it is reasonable to expect a response within a defined period of time. Two global directives are provided to set and test a timeout value in order to prevent a script from remaining totally inactive while waiting for a response not forthcoming. The timeout interval can be set by the directive shown in figure 2-8.

```
=TIMEOUT IS  nnnn
```

nnnn — The number of decimal seconds, 0 to 4095, to wait after each send, before timing out.

Figure 2-8. TIMEOUT Directive Format

The TIMEOUT directive sets a timer that runs from the time a message is sent until the number of seconds expires or a response is received. The timer is reset automatically each time a message is sent from the terminal. The timeout value can be changed at any point in the script where a different response time is anticipated.

Whenever the timer reaches zero, a timeout occurs.

A nonimmediate conditional directive can be specified in conjunction with the TIMEOUT IS directive so that, if timeout occurs, SCRIPT execution will transfer to another location in the same script defined by a label. The directive for this transfer is shown in figure 2-9.

```
=ON TIMEOUT JUMP  TO  eeeeeee
```

eeeeeee — Script location defined by a LABEL directive.

Figure 2-9. ON TIMEOUT Directive Format

If a timeout occurs before an ON TIMEOUT directive is executed, the terminal affected will terminate with the message NO TIMEOUT JUMP ADDRESS. If no TIMEOUT IS directive is in effect, the timeout value is infinite; timeout will never occur.

A declared =TIMEOUT IS nnnn can be cancelled with a =TIMEOUT IS 0 directive.

GLOBAL WAIT FOR RESPONSE

This directive provides a means of suspending script execution after each upline MSG block transmission until a MSG or BLK type block response is received from the host system. This method of operation is similar to step mode operation. Only one message at a time will be outstanding. The format is shown in figure 2-10.

=WAIT { GLOBAL / GLOBAL MSG / GLOBAL OFF }

GLOBAL	This option specifies script suspension after each SEND until either a BLK or MSG reply is received.
GLOBAL MSG	This option specifies script execution suspension after each SEND until a MSG reply is received.
GLOBAL OFF	This option cancels the global wait declaration.

Figure 2-10. Global WAIT Directive Format

GLOBAL SCRIPT THINK-TIME

Timing aspects of a script, such as operator think time, can be controlled with the DELAY GLOBAL directive. This directive provides a delay to be invoked just before each upline transmission to give the effect of user think time. The format is shown in figure 2-11.

=DELAY GLOBAL { nnn / INTERVAL nnn TO nnn }

nnn Decimal seconds, 0 to 1023.

Figure 2-11. DELAY GLOBAL Directive Format

When the first form of this directive is encountered, script execution is suspended for nnn decimal seconds before each message is sent upline. In the DELAY GLOBAL INTERVAL form, a different delay time is selected between the two values of nnn for each send. The sequence of delay times selected for each run is repeatable. The values must be different and the lower value must be specified first. A =DELAY GLOBAL 0 will turn the delay off.

An alternate form of the DELAY is available as a STIM input statement which affects all scripts instead of just the one in which it is specified.

PROCEDURAL DIRECTIVES

Message dialog with a predefined sequence can be controlled through checking of response messages by the use of conditional directives. These directives are used as in-line verification procedures, called local procedures. The conditional directives are used to verify/check either batch or interactive mode response messages received from the host. NPS assumes that all terminals operate with the ASCII character set.

CONDITIONAL DIRECTIVES

The following conditional directives, used separately or together, constitute a verification procedure.

Line Count Verification

The script writer can use the IF LINES directive to determine the approximate length of a message and cause execution to transfer to the specified location if the condition is satisfied. The format of the IF LINES directive is given in figure 2-12.

=IF LINES { LE / GE } nnn JUMP TO eeeeeee

LE	Less than or equal.
GE	Greater than or equal.
nnn	Decimal number of lines, 1 to 511.
eeeeeee	Script location defined by a LABEL directive.

Figure 2-12. IF LINES Directive Format

The number of logical lines of output per message are counted and compared with nnn. The test is applied to each block of output. The line count is accumulative for a message consisting of multiple output blocks. Depending on the operation chosen, less than or equal (LE) or greater than or equal (GE), a determination is made as to whether the condition is satisfied.

When the GE option is used, the first block of data received is tested against the line count. If the volume of data is great enough to satisfy the test, script execution transfers to location eeeeeee, which must be defined as a label. If the volume is not sufficient, lines are counted from subsequent blocks until either the test is satisfied and the jump to eeeeeee taken or until the end-of-message is received. If the test fails at the end-of-message, the next script directive is executed.

When the LE option is used, the test condition cannot be satisfied until the end-of-message is received. If the data received at the end of the message satisfies the test condition, the accumulated line count is zeroed and script execution transfers to location eeeeeee. If the data received is greater than the line count specified, the next script directive might be executed before the end-of-message is received.

Only the first block of a received message is matched. If the received message is made up of several BLKs, the next script instruction might be executed before the end-of-message is received whether the match succeeds or fails. (This also can occur on =IF LINES if the LE fails or the GE succeeds prior to receiving the MSG block). If this occurs, the scripts will probably get out of step. The user must ensure that loss of synchronization does not occur. One method to prevent such an occurrence is to include extra =WAIT MSG directives where a problem might arise.

For both the GE and LE options, the next directive to be executed is checked prior to execution to see if it is an IF LINES directive. If it is, the line count is not zeroed before the next directive is executed. Multiple IF LINES directives can be used to transfer execution to different locations, depending on the length of the message.

Message Verification

The IF MATCH directive can only be used in interactive mode. This directive allows the script writer alternative methods of character-by-character comparison between the response from the host software/application and a message embedded in the script, or defined by a CMESSAGE. The format of the directive is shown in figure 2-13. Any or all of the options can be specified, but they must be specified in the order MASK, LINE, WITH.

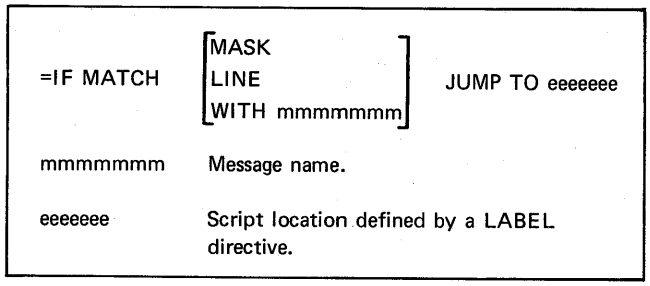

Figure 2-13. IF MATCH Directive Format

If the WITH mmmmmmm option is specified, the message name mmmmmmm must be specified in a =CMESSAGE mmmmmmm script directive in a script of the same TIP-type. If the WITH mmmmmmm option is not specified, the message text must be embedded in the script immediately following the IF MATCH directive.

The MASK option specifies where the character comparison is to start. If MASK is not specified, comparison starts with the first character of both the response message text and the compare message text; that is, the first character of the response is compared with the first character of the message in the script. If MASK is specified, the mask message is limited to 72 characters in length; the mask itself is constructed by the script writer by inserting the character ≠ (CDC display code character 64_8) in those character positions to be ignored in the comparison. Only one mask per line is allowed. For example, a mask message of ≠ ≠ ≠ TERMINAL NAME would seek to find TERMINAL NAME in character positions 4 through 16 of the response message text. If a match is obtained, execution transfers to location eeeeeee, which must be defined as a label. If a match is not obtained, execution continues with the next script directive.

The length of the base message, or mmmmmmm, determines the number of characters that are compared. If the response message block text is longer, its extra characters are ignored; if the response text is shorter, a match does not occur. If the two message texts are the same, execution transfers to location eeeeeee, which must be defined as a label. If the message texts do not match, execution continues with the next script directive.

If the LINE option is specified, a mask or nonmask comparison can be performed as described above. When LINE is specified, the comparison is performed on each logical line in the received block. If none of the lines in the block match, execution continues with the next script directive. If a match is found, execution transfers to location eeeeeee, which must be defined as a label. If the LINE option is not specified, a mask or nonmask comparison is performed; however, the comparison is performed only once on the received block starting with its first data character.

MODE SYNCHRONIZATION

The script writer must declare the mode in which the script is to function; the initial mode of every script is interactive. The MODE directive places the script in batch or interactive operation mode. The format is shown in figure 2-14.

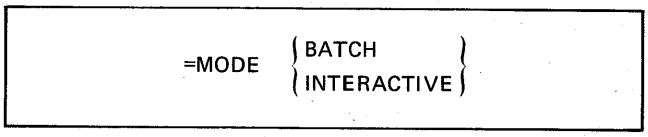

Figure 2-14. MODE Directive Format

On some terminals the resources are switched between supporting batch devices and interactive consoles; such is the case with the Mode 4 (200UT) batch terminals.

The MODE directive must be used to synchronize the mode of the script with the mode of the device currently being simulated by the script. That is, when executing a SEND directive, the mode of the script must correspond to the mode of the device performing the send.

All upline messages are compiled to the format specified by the SEND directive, which is consistent with the MODE directive. This ensures proper format and synchronization when response messages are subsequently referenced in comparison with match messages.

The MODE BATCH directive places the script in batch mode. This allows simulation of a batch terminal device, such as a card reader. All script directives following MODE BATCH assume batch mode.

The MODE INTERACTIVE directive places the script in interactive mode. This allows simulation of an interactive device, such as a console. All script directives following MODE INTERACTIVE assume interactive mode.

The following relates how the =MODE directive affects certain other directives in order to control and synchronize the script and device data:

- The mode of the script at the time the conditional IF directives are encountered must be the same as the mode of the message response received from the host system for the IF condition to be executed; therefore, the correct MODE directive must precede the desired mode of the IF directive.
- The =XSEND directive simulates transmitting data from a batch or interactive device; therefore, the appropriate MODE directive must precede the =XSEND directive.

LOCAL PROCEDURES

A local procedure is an in-line sequence of directives and consists of one or more sequentially executed IF directives. Local procedures are executed only when they are detected during sequential script execution.

When a local procedure is encountered, script execution will wait until a data block is received from the host or until timeout occurs. When the data block is received, the verify procedure is executed. Multiple IF LINES directives can be used to examine the same message data received from the host. Multiple IF MATCH directives are executed independently.

DATA MESSAGE INPUT

Terminal-originated messages are transmitted to the host system via some form of the SEND directive. These messages were originated by the script writer in one of the following ways:

- Defined by an XMESSAGE directive in the script and referenced by a SEND directive.
- Created by the user as a local file and referenced by a SEND directive.
- Defined by embedding the message text within the script immediately following the SEND directive.

Further discussion of the first two sources of message files can be found under Message Definition earlier in this section.

SEND DIRECTIVE

The SEND syntax is the root of the SEND directive. There are several derivatives of SEND. The interactive SEND allows the user to impose control on the SEND operation by the presence or absence of a C character suffixed to the SEND syntax. Additionally, each SEND root is prefixed with an I or B character to designate whether the send is an interactive or batch SEND directive. The mmmmmmm parameter on all SEND directives is optional. If mmmmmmm is not used on a SEND directive, the message text is assumed to be embedded within the script immediately following the SEND directive. The various derivatives of the SEND directive are described below.

Send Login

This special directive can be used as an alternative to the normal login procedure. Its chief purpose is to allow the user to obtain the user name from the TERM input statement rather than specify it in the message text.

The format of this special directive is:

```
=ISEND LOGIN  mmmmmmm
```

This directive results in a modified version of the message named mmmmmmm, defined in an IMESSAGE directive, or the embedded text following (if mmmmmmm is not specified) being sent as the terminal input to the host system. The message text must include seven characters which will be replaced by the username from the TERM input statement. This is often useful when the user name is not known at the time the script is written, or when multiple user names are desired for use with one script as described under option 1 of the TERM Statement in section 3. An example of this usage is shown in figure 2-15.

```
=IMESSAGE MESSG1
SYS172,#######,SESAME,RBF≤H
=ISEND LOGIN MESSG1
```

SYS172	FAMILY prompt response.
#######	Mask for user name.
SESAME	PASSWORD prompt response.
RBF	APPLICATION prompt response.
MESSG1	Message name.

Figure 2-15. Abbreviated Login Procedure Example

The ISEND LOGIN directive can be used with either the full or the abbreviated login procedure. For the abbreviated login procedure, the user can respond to the FAMILY prompt with a message that includes the text for the FAMILY, mask for user name, PASSWORD, and APPLICATION prompts as follows:

```
=ISEND LOGIN
SYS171, ####### ,SECRET,RBF≤H
```

For the full login procedure, the response for the user name can be submitted as only the seven # character mask as shown in figure 2-16.

```
=ISEND
SYS171≤H
=WAIT MSG
=ISEND LOGIN
#######≤H
=WAIT MSG
=ISEND
```

Figure 2-16. Full Login Procedure Example

For the full login procedure, a combination of =ISEND and =ISEND LOGIN must be used to respond to each of the prompts. The =ISEND LOGIN directive must be used to respond to the USER prompt; responses to all other prompts must use =ISEND directives.

NOTE

Logout is performed by use of an ISEND directive sending a user-defined message as specified in the script, followed by statements that subsequently account for any responses from the host system.

Send Message

This directive is the simplest form of SEND and results in a message, defined by a MESSAGE directive or an embedded script message, being sent as terminal input to the host. The format of the SEND directive is shown in figure 2-17.

=XSEND [mmmmmmm]	
=BSEND	Indicates batch mode.
=ISEND	Indicates interactive mode.

Figure 2-17. SEND MESSAGE Directive Format

Send Message Class

This directive allows the script writer to designate a class or identifier for messages being sent to the host. Interactive commands being communicated to an application can be identified by a class designator, which permits specific interactions between the host application and NPS to be measured by postanalysis procedures of the REPORTR module of NPS. The class designator is a positive integer with a range of 1 to 25.

The format of the send message class directive is shown in figure 2-18. This directive causes the message named mmmmmmm, or the embedded text following, to be designated as class type cc and sent as a terminal input to the host. The next interactive response received is automatically designated as the same class.

=ISENDC [mmmmmmm] cc	
cc	Class type, decimal value, 1 to 25.

Figure 2-18. SEND MESSAGE Class Directive Format

Send From lfn

This directive allows the script writer to define messages on a local file rather than defining them in the script. The user-defined lfn file is processed by the SCRIPT compiler and the message text is included in the script library. The lfn must be a legal system logical file name and must be local during execution of SCRIPT. SCRIPT performs automatic blocking on the file. A special character ≤I results in transmission of a BLK block. A special character ≤H results in transmission of a MSG block for interactive data. An =EOR or a system defined end-of-record results in transmission of a BLK block. An =EOI or a system defined end-of-file or end-of-information results in transmission of a MSG block for batch data. The file is rewound each time it is read. The same lfn can be used by more than one script TIP-type. The format of the SEND FROM lfn directive is shown in figure 2-19.

=XSEND FROM lfn	
=BSEND	Indicates batch mode.
=ISEND	Indicates interactive mode.
=ISENDC FROM lfn cc	
cc	Class type, decimal value, 1 to 25.

Figure 2-19. SEND FROM lfn Directive Format

Data characters of the messages generated from the user-created sequential file lfn are serially sent as the text of a terminal input. All messages are sequentially read and transmitted in block form to the host system. The rate at which the individual messages are transmitted to the host system is controlled by the Global Wait for Response and the Global Script Think-Time directives described earlier.

MISCELLANEOUS FUNCTIONS

The following directives allow the script writer additional flexibility in simulating realistic terminal sessions. The WAIT directives are similar to the WAIT GLOBAL directive but are executed only once.

Wait MSG Response

To control the terminal workload described by the script and thus the execution of the script, it is sometimes necessary to determine various positions of the output data stream received from the host system. The WAIT MSG directive causes execution of the script to be suspended until a MSG block is received:

=WAIT MSG

Wait Next Response

To control the terminal workload described by the script and thus the execution of the script, it is sometimes necessary to determine the occurrence of various message responses received from the host system. The =WAIT directive causes execution of the script to be suspended until either a MSG or BLK block response is received from the host system. The format is:

=WAIT

IVT Terminal Commands

The IVT directive provides a means of simulating terminal user-entered IVT (interactive virtual terminal) commands. (Refer to the Network Access Method Reference Manual for characteristics of these commands.) The format of the IVT directive is shown in figure 2-20. Only the commands which cause information to be sent to the application in the host computer are provided; other commands are not provided.

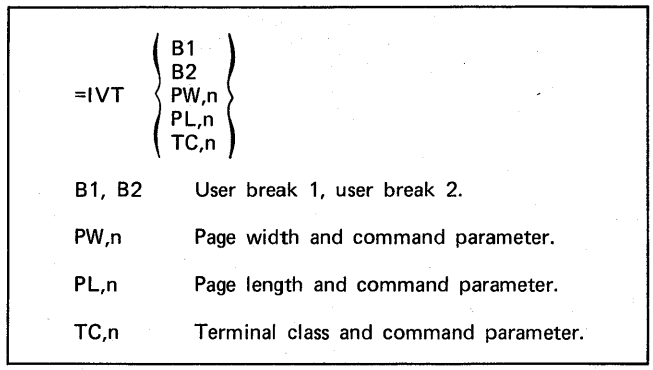

Figure 2-20. IVT Directive Format

Unconditional Branch

The GO TO directive provides for an immediate change of location of execution in the script; for example, it might be used to prevent fall-through in a sequence of conditional statements:

```
=GO TO eeeeeee
```

This directive causes an immediate transfer of execution to location eeeeeee, which must be defined as a label.

Execution Loops

Execution of various sequences of script directives can be repeated for a specified number of times by including the REPEAT directive as the last directive in the sequence. The format is shown in figure 2-21.

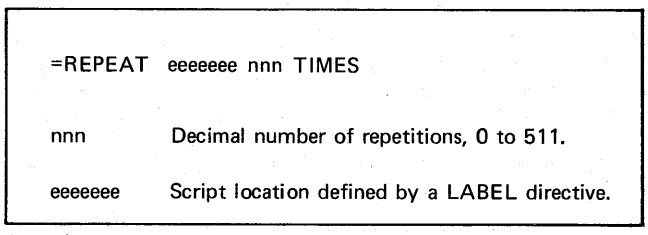

Figure 2-21. REPEAT Directive Format

The REPEAT directive causes a counter to be set that is decremented by one for each subsequent time this directive is encountered. Execution then transfers to the label eeeeeee, as long as the counter is nonzero. When the counter reaches zero, REPEAT is nullified and execution falls through to the next directive.

When script execution is transferred outside of a REPEAT range, as a result of the execution of any branch statement (for example, =IF, =ON TIMEOUT, =GEJ, =GOTO), the counter also becomes zero.

When the REPEAT directive is encountered, it enables the sequence of instructions, executed once prior to the REPEAT directive, to be executed nnn additional times. The label eeeeeee must be declared prior to the REPEAT directive. Three levels of nesting are allowed on REPEAT loops. Any error on a REPEAT statement causes SCRIPT to bypass syntax checking on subsequent REPEAT statements.

Dayfile and Log Message from Script

Error conditions, status statements, and so forth, can be recorded during script execution through use of the LOG directive:

```
=LOG text
```

The LOG directive causes text to be written to the Stimulator log file and the STIM job dayfile. Terminal and script identifiers are output with the message. The text must be 60 characters or less in length. If longer, the text is truncated to 60 characters. The text begins with the sixth character position in the line.

Termination of Script Execution

Normal termination of script execution is accomplished by the EXIT directive:

```
=EXIT
```

Script execution stops, and the associated terminal becomes idle.

Network Hardware Failure Simulation

Two directives allow the script writer to simulate failure and automatic recovery hardware conditions that can occur in the network during terminal connection to an application.

The line failure directive causes a supervisory message to be sent to the host signaling that the line being used by the terminal or script has failed. The format is:

```
=LFAIL
```

When the =LFAIL directive is executed, the line is recovered and script execution continues. The next data message received is the same as that received when the line first became operational.

The terminal failure directive causes a supervisory message to be sent to the host signaling that the terminal has failed. The format is:

```
=TFAIL
```

When the =TFAIL directive is executed, the terminal fails and is recovered in the same manner as if it had been turned off and on again.

PROGRAM CONTROL

Two types of program control are available to the script writer. Event recognition can be used to suspend execution of a script until a predetermined action has occurred in that script or in another script. Events can be declared and used to control the execution of more than one script; this is called interscript control. Event declaration can be used to coordinate multiple HASP scripts.

Within a script, counters can be declared, and their values can be manipulated in order to control execution in a script. Counters do not allow interscript control.

Event Declaration

The EVENT directives are the only script directives that provide interscript communication. An EVENT directive can be used to cause a script to suspend the sequential execution of script directives until the declared event is set on by another script.

The maximum number of declared events (events with a nonzero value) in a STIM run is limited by the assembly option variable MAXNEVTS; default value of MAXNEVTS is 60. The maximum value of MAXNEVTS is limited to 128. Execution of EVENT directives greater than MAXNEVTS will result in a diagnostic dayfile message and termination of the stimulation run during initialization with an informative dayfile message. Declared events in any script are global to all scripts.

Event Recognition

The EVENT ON directive indicates that the event vvvvvv has occurred and sets the event to an on state. Execution of an identical =EVENT ON vvvvvv directive for an event already on increments the event cell by one; in this way the event declarations can, in addition to recognizing on and off states, accumulate a count of similar event occurrences. The range of the count is 0 to 1023, with 0 being the off state. The format is:

```
=EVENT ON vvvvvv
```

Event Waiting

The EVENT WAIT directive causes script execution to wait until the event vvvvvv has been set to the on state before the next sequential script directive is executed. Multiple EVENT ONs can be balanced by multiple EVENT WAITs. If the event has occurred at least once (count is >0) the count is decremented by one and the next directive is executed. If the event has not occurred or the count has already been decremented (count is =0), script execution waits until the event occurs (count > 0). The format is:

```
=EVENT WAIT vvvvvv
```

If multiple scripts are waiting for the same event, only one script will be allowed to advance for each recognition of the event (EVENT ON directive executed).

Any data sent downline to a terminal that is waiting for an event to occur will not be processed.

Counter Setting and Checking

The following directives allow the script writer to set and check integer counters, which can be used to further control terminal sessions. These counters are local to a terminal; they do not provide interscript communication as do the EVENT counters. The maximum number of counters per terminal is six.

Set Counter

The SET directive declares a name, ccccccc, of up to seven characters as an integer counter and sets it to the integer value specified by nnn, where 0 ≤nnn ≤511. The format is:

```
=SET ccccccc,nnn
```

If the counter is not initially set, further operations described below will not be trustworthy.

Increment Counter

The INC directive increments a counter named ccccccc by the integer value m, where 1 ≤m ≤9. The format is:

```
=INC ccccccc,m
```

Decrement Counter

The DEC directive decrements a counter named ccccccc by the integer value m, where 1 ≤ m ≤ 9. The format is:

```
=DEC ccccccc,m
```

Compare Counter

The GEJ directive compares the value in the counter named ccccccc to the integer value mmm (0 ≤ mmm ≤511). If the counter value is greater than or equal to mmm, execution transfers to location eeeeeee, which must be defined as a label. If the counter is less than mmm, execution continues with the next directive. The format is:

```
=GEJ ccccccc,mmm,eeeeeee
```

Stimulator operation encompasses compilation of user-written scripts by the SCRIPT compiler and stimulation of the network products software using the library of compiled scripts. Input and output is time-stamped and recorded on a log file for subsequent analysis.

SCRIPT COMPILER

The SCRIPT compiler is called by a control statement from the system library and runs as a normal batch job. The parameters on the call statement are used to specify the input/output options desired.

The initialization of the SCRIPT compiler includes reading the control statement options and modifying the default options accordingly.

The scripts are read from the specified input file. Nonembedded messages from the scripts are written to the message part of the script library file. Script directives are converted to numeric representation. Listings of scripts are generated if requested. Finally, the second part of the new script library file is written containing the converted scripts in run-time format.

SCRIPT CALL STATEMENT

The SCRIPT program call statement has the format shown in figure 3-1. N, I, LO, and L are keyword parameters that assume the values described in the figure. The parameters are order independent. Defaults are assumed if the parameters are omitted or specified without values.

Errors in the control statement parameters cause the run to be terminated with diagnostic error messages. The system file name ZZZZZS1 is reserved for use by the SCRIPT compiler. All nonembedded messages are compiled together, so, on the output listing, messages are not necessarily associated with the scripts in which they are referenced.

The following are examples of SCRIPT call statement use:

SCRIPT.
Scripts are read from INPUT and the script library NEWSL is created. Only fatal errors are listed on the file OUTPUT.

SCRIPT(N=SL12345,I,LO=F)
Scripts are read from the file COMPILE, creating the new script library SL12345. All scripts and all messages are listed on the file OUTPUT.

SCRIPT(N=nsl,I=ifn,L=ofn,LO=opt)

nsl Logical file name of the new script library created by this run. The default is NEWSL.

ifn Local input file name from which the scripts are read:

If =ifn is omitted, input is on file COMPILE

If I is omitted, input is on file INPUT

ofn Local output file name on which the requested listings are generated.

If =ofn is omitted, listable output is on file LIST

If L is omitted, listable output is on file OUTPUT

If L=0, no output file is written, and listing options specified are ignored

opt Listing options. Only one option can be specified; the default is fatal errors only. The options are:

E	Only fatal errors are listed; this is the default value
F	All scripts and all messages are listed
M	All nonembedded messages are listed, as well as the scripts specified on LIST directives
Q	Only the scripts specified on LIST directives are listed without nonembedded messages
S	All scripts are listed without nonembedded messages

Figure 3-1. SCRIPT Call Statement Format

SCRIPT DECK STRUCTURE

The example shown in figure 3-2 illustrates a SCRIPT compiler run. It updates an old program library and generates a new script library. The files reside on mass storage.

Notice that an Update slash (/) program library is used in the example. It is recommended that the user maintain a special character program library so as not to restrict the format of messages or jobs to send upline.

```
JOB.
USER,RAC,SESAME,1234.
ATTACH(OLDPL=OLDPL5)
DEFINE(NEWPL=NEWPL6)
UPDATE(F,N,*=/)
DEFINE(NEWSL=NEWSL6)
SCRIPT,LO=S,I.
7/8/9
/ADDFILE COMCOM
/COMDECK COMLOG
=COMMENT STANDARD LOGIN PROCEDURE
=ISEND LOGIN
  .
  .
  .
/ADDFILE ,,SCRIPTX
/DECK SCRIPTY
=SCRIPT SCY
  .
  .
  .
/CALL COMLOG
  .
  .
  .
=ENDSCRIPT
/IDENT EXAMPLE
/DELETE . . . (further modifications)
  .
  .
  .
7/8/9
6/7/8/9
```

Figure 3-2. SCRIPT Compiler Run Example

SCRIPT FILE FORMATS

The script library consists of two parts:

Part 1 - script messages. This part consists of all nonembedded data messages that are defined in the scripts in part 2. Each message is one logical record.

Part 2 - script object code. This part consists of the compiled object code for all scripts. Each script occupies one logical record and a directory record is appended at the end.

SCRIPT ERRORS AND TERMINATION

SCRIPT compiler processing falls into two main categories, an input pass and a binary pass. In the input pass, SCRIPT directive keywords and XMESSAGE directives are checked. Errors in these directives terminate the SCRIPT run, and the message ERRORS IN INPUT - RUN ABORTED appears in the dayfile.

If no errors are found in the input pass, the binary generation pass is then executed. Here, SCRIPT performs all syntax checking and ensures that each script and the script library as a whole are complete and compatible (for example, it ensures that all entries and messages used have been declared). Errors in this pass terminate the SCRIPT run with the dayfile message ERRORS IN BINARY - RUN ABORTED.

Upon abnormal termination, all statement images found to have syntax errors appear in the job output file unless the L=0 option has been declared. Normal end-of-run is indicated by the dayfile message SCRIPT COMPLETE.

STIM

The central processor and peripheral processor stimulation routines, STIM and SIP, are used to supply input to and receive output from NAM and the application using NAM.

The central processor program STIM is initially loaded by a program call statement, for example, STIM(n_1,n_2,...n_n), that specifies the driver and monitor parameters. Then STIM interprets and saves these parameters for later use in determining the characteristics of the stimulator run.

STIM then reads the LNODE, RNODE, HASPCR, HASPLP, DELAY, and TERM input statements, which specify terminal parameters. From these input parameters, STIM builds the communication tables. STIM then initializes its central memory buffers, builds a communication buffer for the peripheral processor program SIP and requests SIP to be loaded. SIP reads the communication buffer, initializes its peripheral processor memory, and then signals STIM that the peripheral processor is loaded and initialized.

STIM then completes its initialization pass by configuring the network and enters the simulation pass where script execution is performed. At this transition, the system real-time clock is read to establish a relative time zero. All statistics generated by REPORTR are relative to this zero time.

Once stimulation is in process, STIM's function is to supply input from simulated terminals (scripts) to SIP and to receive from SIP all output messages directed to simulated terminals. STIM, if so directed by the LOGGING IS directive, records all input and output messages on a log file. During this time, SIP communicates with the NAM peripheral processor routine PIP just as the NPU does, sending and receiving messages across the dedicated data channel, and emulating the hardware and software control functions of the NPU. The fact that this NPU is simulated rather than real is transparent to PIP and the entire network.

Characteristics of the stimulation run are controlled by parameters on a STIM program call statement. Parameter checking is performed on the statement, and any errors result in termination of the run with diagnostic dayfile messages.

STIM CALL STATEMENT

The STIM program call statement has the format shown in figure 3-3. SL, LF, ET, I, LC, and NC are keyword parameters that assume the values described in the figure. The parameters are order independent.

INPUT STATEMENTS

During initialization, parameter checking is performed on the input statements that follow. Any errors result in termination of the run with diagnostic dayfile messages.

STIM(SL=sln,LF=lfn,ET=nnnn,I=ifn,LC=lcf,NC=ncf)

sln Logical file name of the direct access file containing scripts (that is, the script library file generated from a SCRIPT compiler run); one to seven letters or numbers. This parameter can be omitted; the default is NEWSL.

lfn Logical file name on which data traffic is to be recorded; one to seven letters and numbers. This parameter can be omitted; the default is LOGNPS.

nnnn The decimal number of seconds representing the time limit of the simulation test; STIM terminates after nnnn clock seconds have elapsed or upon exit of all scripts. This parameter can be omitted; the default assumes an infinite time period. The legal range is $0 < nnnn \leq 28800$ (=8 hours).

ifn Local input file (one to seven letters or numbers) name on which STIM input statements (such as TERM, LNODE, and so forth) are expected to reside. The default is INPUT.

lcf Logical file name of the direct access local configuration file; one to seven letters or numbers. This parameter must not be omitted; if omitted, the run is terminated with an appropriate diagnostic dayfile message.

ncf Logical file name of the direct access network configuration file as defined by NDL; one to seven letters or numbers. This parameter must not be omitted; if omitted, the run is terminated with an appropriate diagnostic dayfile message.

Figure 3-3. STIM Call Statement Format

Think-Time Statement

The DELAY input statement serves to establish the same think-time for all interactive scripts (terminals) in a test. The parameter options are the same as those included in the DELAY GLOBAL script directive (described in section 2), but are specified in a different manner. However, where the script directive establishes a think-time applicable only to the individual script in which the directive is executed, this input statement establishes a think-time applicable to all the scripts for an entire test run. Furthermore, this input statement overrides any DELAY script directive established values.

The think-time input statement has the format shown in figure 3-4.

{DELAY / D} (p)

p Assumes one of two formats representing the following corresponding option:

nnn nnn=Decimal seconds 0-1023

I=nnn/nnn

Figure 3-4. Think-Time Input Statement Format

Input Speed Delay Statement

For interactive asynchronous terminals, a delay corresponding to a user's typing rate of 1 through 10 characters per second can be simulated using the IS input statement. The delay occurs prior to sending the upline message to the host and can be used in conjunction with the other delay statements. When the IS input statement is not specified a default typing rate of four characters per second is used for all interactive terminals. An IS value of zero turns the delay off allowing the terminal to simulate an infinite typing rate. The IS cards can occur anywhere among the NPS input statements.

The two formats for the IS input statement are shown in figure 3-5. The first form causes all interactive terminals to input isx characters per second. If more than one IS card of this format is detected, the value in the last IS card will be used. The second form causes the first nt1 terminals to operate at is1 characters per second, the next nt2 terminals to operate at is2 characters per second, and so forth. These values are assigned to terminals in the order specified by the TERM cards. The two IS forms cannot be used in the same NPS run. An example of the second format is:

IS(3/7,1/0,2/2)

This statement would cause the first three terminals to simulate a typing speed of seven characters per second. The next terminal would simulate an infinite typing speed, and the following two terminals would operate at two characters per second. If the run included more than six interactive terminals, the remaining terminals would operate at the default IS value of four characters per second. If the run included fewer than six interactive terminals, the extra IS values would be ignored.

IS {(isx) / (nt1/is1,nt2/is2, . . .ntn/isn)}

where $0 \leq isx \leq 10$

and $0 \leq ntx \leq 1023$

Figure 3-5. Input Speed delay Statement Format

LNODE Statement

This input statement serves to identify which front-end NPU in the local configuration file is to be configured for simulation by STIM. A maximum of four local NPUs can be configured for simulation. Up to four NPUs can be configured; at least one must be a local NPU.

One or more TERM input statements must immediately follow the LNODE input statement to identify the terminals connected to the local NPU. The format of the LNODE statement is shown in figure 3-6.

RNODE Statement

This input statement serves to identify which remote NPU in the local configuration file is to be configured for simulation by STIM. A maximum of three remote NPUs can be configured for simulation. Up to four NPUs can be configured; at least one must be a local NPU.

{LNODE / L} (n)

n A simulated local NPU node ID from the Network Definition Language NPU statement in the local configuration file.

Figure 3-6. LNODE Input Statement Format

One or more TERM input statements must immediately follow the RNODE input statement to identify the terminals connected to the remote NPU. The format of the RNODE statement is shown in figure 3-7.

{RNODE / R} (n)

n A remote NPU node ID from the Network Definition Language NPU statement in the local configuration file.

Figure 3-7. RNODE Input Statement Format

TERM Statement

This input statement has the format shown in figure 3-8. LN, TN, SN, LG, UN, and LS are keyword parameters that assume the values described in the figure. The parameters are order dependent, because the slash (/) is a separator for option 1.

The TERM input statement serves to identify terminals configured in the network local configuration file tables, which are to be selected by NPS for simulation. To achieve this identification, the TERM statement can be used in two ways. Option 1 cannot be used with HASP terminals.

Option 1 - Range of Terminal Identification

This option allows the user to identify a range of terminals in the local configuration file. All terminals within the range, including the boundary terminals, are selected from the local configuration file for simulation. This option makes use of the / separator and two ordered sets of LN and TN parameters. Terminals selected by this option all have the same TERM statement LG and SN parameter characteristics, but the user name can vary. The following is an example of this option:

```
LNODE(3)
TERM(LN=ABC,TN=WW/LN=DEF,TN=ZZ,SN=AAA,
    LG=10,UN=AAA6879)
```

This example illustrates that all lines between ABC and DEF, inclusive, and terminals in the local configuration file between WW and ZZ, inclusive, on these lines are to be simulated. Furthermore, all the terminals are connected to local NPU node 3, are to login after 10 seconds, and are to execute script AAA. The / serves to separate and identify the lower and upper range of the terminal list; therefore, this group of parameters must be ordered in the statement. All other parameters can appear in any order. If the user name is specified, the last four characters must be numeric. The user name is sequentially incremented in decimal for each included terminal in the range for use if an =ISEND LOGIN directive is encountered in the script.

{TERM / T} (LN=linename$_1$,TN=trmname$_1$/LN=linename$_2$,
TN=trmname$_2$,SN=sid,LG=nnnn,UN=user name,
LS=line speed)

linename$_i$ Name of the line from the NDL LINE statement in the local configuration file to which this terminal is connected. No validity check is made against the local configuration file for this name. The first LN parameter cannot be omitted; the second LN parameter is optional (described in option 1).

trmname$_i$ Terminal name from the NDL TERMINAL statement in the local configuration file. The first TN parameter cannot be omitted; the second TN parameter is optional (described in option 1).

sid Name of the script on the script library that this terminal will execute; one to three letters or digits. This parameter cannot be omitted.

nnnn Decimal number of seconds representing this terminal's login time. This time is relative to the zero time or the start of the simulation test run; this terminal will login no sooner than nnnn seconds after time zero. Login is defined to be the time when the line this terminal is on is activated. Parameter LG is optional. The default is zero. The legal range is $0 \leq nnnn \leq 3600$.

user name Optional terminal operator's valid Network Operating System user name; one to seven letters or digits. If the range of terminal identification (option 1) is used as described, the last four digits of the user name are sequentially incremented for each succeeding terminal in the range. This parameter need be included only if the script (sid) contains an ISEND LOGIN directive.

line speed Optional line speed simulated for the terminal. Legal values for asynchronous lines are 110, 150, 300, 600, 1200, 2400, 4800, and 9600 bps. The default value is 300 bps for auto-speed recognition lines. LS values greater than the legal auto-speed maximum of 1200 bps will be honored although an informative diagnostic will be issued. For non-auto-speed recognition lines, the default line speed is provided by the host from the line configuration file.

Legal values for synchronous lines are 2000, 2400, 4800, 9600, 19200, and 50000 bps. The default value is 2400 bps.

A line speed value of zero causes NPS to simulate an infinite line speed.

Figure 3-8. TERM Input Statement Format

In this example, the next username would be AAA6880, then AAA6881, and so on through AAA9999, and then becomes AAA0000.

Option 2 - Individual Terminal Identification

This option allows the user to identify a single terminal in the local configuration file to be selected for simulation. This option does not use the / separator; only one set of LN and TN parameters is used. The following is an example of this option:

```
LNODE(3)
TERM(LN=ABC,TN=WW,SN=AAA,LS=2400)
```

This example illustrates that terminal WW on line ABC, connected to local NPU node 3, is to be simulated. Execution of script AAA will begin after time zero. No username will be supplied from the TERM card. NPS simulates a line speed of 2400 bits per second for this terminal. This causes a delay in the delivery of upline batch input messages as well as a delay in the acknowledgment of downline batch and interactive output messages. The delay corresponds to the time required to receive or deliver the data and depends on the message length and the line speed.

The relationship of TERM input statements to MODE4, ASYNC, and HASP terminals is noted in section 5.

HASPCR Statement

The HASPCR input statement serves two purposes. It identifies the scripts that are to be used to simulate HASP card reader input, and also identifies the number of card readers to be simulated on a particular HASP station. Examples of HASPCR use appear in section 5.

If a HASP card reader is to be simulated, the HASPCR input statement must follow the TERM input statement defining the console display for the station. The format of the HASPCR statement is shown in figure 3-9.

{HASPCR / HC} (sss_1, . . ,sss_n)

sss_i The name of the script used to simulate a HASP card reader workload; a maximum of seven script names is allowed. The number of script names determines the number of card readers to be simulated. At least one script name must be specified or the run will terminate with an error message during initialization.

Figure 3-9. HASPCR Input Statement Format

HASPLP Statement

The HASPLP input statement serves two purposes. It identifies the scripts that are to be used to process output file data received from the host system, and identifies the terminals that are to be configured and expected to receive the output. It also explicitly associates each script with a particular terminal so that output to a terminal is processed by the associated script.

If any HASP output terminals are to be simulated, the HASPLP input statements must follow the HASPCR input statement defining the card readers for the station. The format of the HASPLP statement is shown in figure 3-10.

{HASPLP / HL} sid_1=$trmname_1$, . . . ,sid_n=$trmname_n$,sidx/N)

sid_i Name of the script used to process the output received from the host system for a HASP terminal; a maximum of seven script names can be used. This parameter is optional; default assumes no output processing and output is merely logged to the log file.

$trmname_i$ Terminal name from the NDL TERMINAL statement in the local configuration file. Output to this terminal is processed by script sid_i.

sidx Name of the script used to process output for all terminals not explicitly associated with scripts. This parameter is optional; if missing, output that is not explicitly associated with a script is not processed, it is merely logged to the log file.

Only one sidx parameter usage is allowed per HASP station simulation, regardless of the number of HASPLP statements used per station.

N Number of terminals assigned to process output with the script sidx. N is a decimal number between 1 and 7. The N parameter is optional; if missing and the parameter sidx is used, N defaults to one. If N is supplied, the sidx parameter must have been specified.

The number of terminals simulated (specified by sid=trmname and by N) cannot exceed seven.

Figure 3-10. HASPLP Statement Format

The following are examples of the legal use of the HASPLP input statement:

HASPLP(XYZ=ABCD)
Script XYZ is to process output received by the terminal named ABCD. Output delivered to other terminals is only logged.

HASPLP(XYZ=ABCD,BBB)
As in the first example, script XYZ is to process output received by terminal ABCD. In addition, the script BBB is to process all output delivered to other terminals.

STIM DECK STRUCTURE

STIM is initiated by executing a job containing the STIM call statement and the LNODE, RNODE, and TERM input statements. The script library and network definition files must be attached before the STIM call card. STIM controls the reading and processing of the input statements to perform its initialization pass. The information necessary for STIM to configure the simulated network is contained in the input statements and in the network configuration files.

The simulated network can use real or imaginary hardware, but must be defined in the local configuration file generated by NDL. The simulated network can be either a subset or the full set of the local configuration file. STIM expects the input statements to be ordered to associate NPUs and terminals. LNODE and RNODE statements must be followed by TERM statements identifying those terminals connected to the specified NPU.

STIM can assume two different job deck structures, one for a batch origin and the other for a system origin job.

A batch STIM job deck is similar to a normal user job containing a control card record followed by an input record. A sample batch job deck structure is shown in figure 3-11.

```
JOB.
USER,DEL0077,SESAME,SYS172.
ATTACH(NEWSL=SL123)
DEFINE(LOGNPS)
ATTACH(NCF=NCF1,LCF=LCF1)
STIM(LC=LCF,NC=NCF,ET=600)
EXIT.
DMP.
DMP(0,10000)
7/8/9
LNODE(parameters)
TERM(parameters)
  .
  .
  .
TERM(parameters)
RNODE(parameters)
TERM(parameters)
  .
  .
  .
TERM(parameters)
7/8/9
6/7/8/9
```

Figure 3-11. Batch STIM Sample Deck Structure

A system origin STIM job has its control statement record and input statements residing in separate files. The control statements must be on an indirect permanent file that uses the system user index 377777_8. The name of this file must start with the three characters, STM, followed by any four characters or digits. The file containing the input statements can be a public file under any user. This file must be attached before the STIM call statement and specified in the statement with the I parameter.

A system origin STIM job is initiated by a console type-in of the indirect permanent file name, STMxxxx. A sample deck of the job is shown in figure 3-12.

STIM OPERATING PROCEDURES

The following operational conditions are necessary for running STIM:

1. Equipment status table (EST) entries of the NPUs being simulated must initially be logically on in order to be identified and used by SIP.

2. The PIP and SIP channels are dedicated.

```
STMXXXX Indirect Permanent File

ATTACH(NEWSL=SL123/UN=USERXXX)
ATTACH(NCF=NCF1,LCF=LCF1/UN=USERXXX)
ATTACH(INFILE=INSTMTS/UN=USERXXX)
RFL(70000)
STIM(LC=LCF,NC=NCF,I=INFILE)
EXIT.
DMP.
DMD(0,40000)


INSTMTS File

LNODE(parameters)
TERM(parameters)
  .
  .
  .
RNODE(parameters)
TERM(parameters)
  .
  .
  .
TERM(parameters)
```

Figure 3-12. System Origin STIM Sample Deck Structure

3. The NPUs simulated must have unique equipment numbers.

4. All equipment on the PIP and SIP channels must be logically off in the EST. Other equipment on the channels must not be turned on in the EST; otherwise, the system will probably crash.

5. If a channel is lost during PIP/SIP correspondence, then all equipment simulated on that channel is lost to the NPS run.

6. Any equipment on the PIP and SIP channels must be physically OFF.

The following procedures assume that deadstart has been completed and all other components of the network system have been initialized.

Initialization

STIM can be initialized at any convenient point after deadstart. Operator procedures are required to ensure the proper interface of NPS to the network host product and the EST entries used for simulation.

A STIM run is initiated by executing a batch or system origin job containing the STIM call statement and the necessary STIM input statements. The first action performed by STIM is to start the initialization pass by informing the user of the version and level of NPS. During this pass, the configuration of channels, PIP, and NPUs is checked for conflicts. If conflicts are present, the run is terminated with appropriate dayfile messages. When a STIM run is initiated by a batch job, the following procedural message flashes at the STIM control point:

AUTHORIZE NPS,xxx,xxx,xxx,xxx

This message requests the console operator to authorize use of the indicated EST entry for NPS use. If two or more users submit STIM runs with identical channels and NPU configuration requests, the operator must determine which job to authorize. If authorization is given, the operator should enter:

n.GO.

If authorization is not given, the operator can enter:

n.DROP., n.RERUN,0., or n.ROLLOUT.

If authorization is given and n.GO is entered by the operator, the following automatic procedures are performed and associated messages are displayed in the STIM user dayfile and the B-display at the STIM control point. A system origin STIM job bypasses the authorization message and action sequence. The next message displayed is:

STIM INITIALIZING NETWORK

Initializing the network consists of receiving and responding to service messages sent from the host, and constructing internal tables that reflect the network configuration to be simulated.

When initialization is completed, the simulation/stimulation test state is entered and is communicated to the operator by the following dayfile message displayed in the STIM user dayfile and on the B-display at the STIM control point:

STIMULATION RUN STARTED

This message is displayed at relative zero time. Relative zero time is defined to be that point where execution of the first script can begin.

Execution

Once the STIM run has been initialized without any errors, the simulation state is entered automatically. That is, all terminals (scripts) will be executed.

Termination

Termination of the simulation test is automatic when all scripts have executed or if an error condition occurs.

Termination of Run by Operator

To terminate the entire simulation run, the operator should enter:

n.DROP. for a batch job

n.STOP. for a system origin job

where n is the control point occupied by STIM.

Termination of Run by Time Expiration

The entire simulation run terminates automatically if the duration of the test, measured in seconds of elapsed clock time, exceeds the time designated on the STIM call statement. In this case, the following message appears on the console A and B-displays:

RUN TIME EXCEEDED. RUN TERMINATED.

Termination of Run by Test Completion

When all scripts have completely executed and all terminals have performed logout and executed an EXIT directive, the entire test automatically terminates and displays the following message on the A and B-display:

STIM RUN COMPLETE

NPS Dumps on Tape

When NPS terminates abnormally and is executed from a system origin procedure file, a network procedure file called NPSDUMP is available to dump the NPS files to tape. NPSDUMP is activated by executing the control statement shown in figure 3-13.

BEGIN,NPSDUMP,NPSDUMP,newsl,lognps,netnps,lcf,ncf.

newsl	Filename of script library. Default is NEWSL.
lognps	Filename of record traffic. Default is LOGNPS.
netnps	Filename of STIM input file. Default is NETNPS.
lcf	Filename of LCF. Default is LCF.
ncf	Filename of NCF. Default is NCF.

Figure 3-13. NPSDUMP Control Statement Format

Any of the parameters may be omitted. The default filename will be what is shown in figure 3-13. The control statement is inserted after the field length dump card, located after the EXIT control statement. NPSDUMP will request a labeled tape. The flashing tape request may be processed by the operator (see NOS Operator's Guide, Network Failure Processing). If NPS fails along with the rest of the network and the NPSDUMP procedure is executed, the NPS dumps are automatically copied to the same tape or set of tapes used for the network dumps. If the network dumps are to be printed, the NPS dumps are also printed.

STIM FILE FORMATS

The input file to STIM is a specified script library file generated by a SCRIPT compiler run. The log file from a run consists of four categories of messages:

- Downline messages from the host
- Upline messages to the host
- Dayfile messages
- Log messages (messages directly from a script)

The purpose of the REPORTR program is to analyze the data resulting from a stimulation run and convert the data into usable information. REPORTR has the capability to reformat the data into a readable form and to collect, compile, and print certain statistical information.

The REPORTR program can perform the following functions, depending on the parameters specified on the REPORTR call statement:

- The recorded event data for a selected time period can be reformatted and printed with the message text.
- The driver and monitor data for a selected time slice can be transferred to a working file and the message text discarded to conserve disk space.
- The contents of the working file can be analyzed and statistical reports generated.

REPORTR CALL STATEMENT

The REPORTR program call statement has the format shown in figure 4-1. LF, D1, D2, T1, T2, DM, R, and L are keyword parameters that assume the values described in the figure. The parameters are order independent. If any keyword=value combination is omitted, a default value is assigned.

Errors in the REPORTR call statement result in termination of the run with a diagnostic error message. The operating system scratch file names ZZZZZS2, ZZZZZS3, and ZZZZZS4 are reserved for use by REPORTR.

The following are examples of REPORTR usage:

REPORTR.

The name of the input log file (LF) is defined as LOGNPS. No message traffic dump is output, and all the data message traffic for the entire run is extracted from the LOGNPS file for analysis. Only the control statement parameter summary is generated.

REPORTR(LF=SIMFILE,D1=600,D2=900,T1=0,
T2=1800,DM=Y,R=3467)

SIMFILE is the name of the input log file. All message traffic occurring between 10 and 15 minutes after the start of the stimulation run is dumped to file OUTPUT. The data analysis spans the first 30 minutes of the stimulation run. The network configuration service messages occurring during network initialization are dumped. Reports 3, 4, 6, and 7 are generated.

REPORTR(LF=lfn,D1=nnnnn,D2=mmmmm,T1=sssss,
T2=ttttt,DM=y,R=ijk . . . ,L=ofn)

lfn	Name of the log file that contains the data generated by the Stimulator. The default file name is LOGNPS.
nnnnn	Starting time of the message traffic dump, specified in decimal seconds. The starting time is defined as the elapsed time into the stimulation run from relative time zero. All messages transmitted and received within this time period are formatted and dumped to the output file. If the dump parameter is omitted, no message dump is output.
mmmmm	Ending time of the message traffic dump, specified in decimal seconds. The ending time is defined as the elapsed time into the stimulation run from relative time zero. If D1 is specified and D2 is not specified, D2 will default to the end of the run.
sssss	Starting time of the data analysis time slice, specified in seconds. The starting time is defined as the elapsed time into the stimulation run from relative time zero. All messages transmitted and received within the time slice are written to the work file after the message text is removed. The data contained on the work file is analyzed and selected reports generated. If the time slice parameter is omitted, the entire stimulation run is analyzed.
ttttt	Ending time of the data analysis time slice, specified in decimal seconds. The ending time is defined as the elapsed time into the stimulation run from relative time zero.
y	A Y value causes the initial service messages issued during network initialization to be printed. If the keyword=value combination is omitted, a default value N is assigned.
ofn	Local file name on which REPORTR output will appear. The default value of ofn is OUTPUT.
ijk . . .	A report type selection parameter that specifies which reports are to be generated. The possible values of ijk . . . are combinations of the integer values 1 through 7 without separators. Integers 1 through 7 correspond to report numbers 1 through 7 (for example, R=2546 would result in reports 2, 4, 5, and 6 being generated).

Figure 4-1. REPORTR Call Statement Format

JOB STRUCTURE

The following illustrates the deck structure of a REPORTR run:

```
JOB.
USER,1234,PASS.
CHARGE,5678,91011.
ATTACH(LOGNPS)
REPORTR(parameters)
7/8/9
6/7/8/9
```

STATISTIC DEFINITIONS

The following statistical terms are used in the reports generated by REPORTR; their interpretation for this purpose is given here.

Arithmetic mean -- The sum of all observations (such as response times) divided by the number of observations.

Range -- The difference between the extreme values in any set of observations.

Standard deviation -- The square root of the variance. The variance is defined as the average squared deviation from the mean.

Interval -- The timeframe in which any number of observations (such as response times) can occur.

Frequency -- The sum number of observations that occur in a particular interval.

Cumulative frequency -- The additional increment of observations found in the next interval in the sequence of intervals.

Percent frequency -- The percentage of the total number of observations found in a particular interval.

Cumulative percent frequency -- The additional increment of the percentage of the total number of cases found in the next interval in the sequence of intervals.

Response time -- The time the host system takes to react to a given input. When a message is sent by a terminal and the reply from the host is received at the same terminal, response time is defined as the time interval between the upline send and the first downline receive. It is the interval between an event and the host system's response to the event. Only those messages which receive a response are considered in response time calculations. Response time is only meaningful for interactive applications.

Total response time -- The sum of the response times for a particular terminal.

Total elapsed time -- The interval starting from relative time zero to the ending time for the stimulation run.

Subset elapsed time -- The sum of the times that a particular number of terminals (subset) is active.

Terminal elapsed time -- The sum of the times that a particular terminal is active.

Active terminal subset -- The particular number of terminals that are logged in but not logged out during any interval of time.

STATISTICAL REPORTS

There are eight statistical reports generated by REPORTR. The first report, which is not numbered, is always printed; it summarizes the control statement parameters. Sort/Merge is used by REPORTR to facilitate computation of statistics included in the reports. Specifically, Sort/Merge is used to generate the optional reports numbered 1 through 7. Additionally, Sort/Merge is used to sort the interleaved message traffic of multiple terminal runs; this results in the capability to list separately and chronologically each terminal's message traffic when the D1 and D2 parameters on the REPORTR call statement are used. Thus, if Sort/Merge is not part of the operating system, none of the reports numbered 1 through 7 can be generated correctly.

The message traffic occurring within the time period specified by the T1 and T2 parameters on the REPORTR call statement is analyzed, and the reports that follow are printed.

The term class in the statistical report refers to the message class, which can have a value of 1 through 25. A class value of zero indicates that a message class was not designated.

The following notation is used in the examples of the statistical reports:

xxxxxxx	will be replaced with the terminal name or blanks.

REPORTR CONTROL STATEMENT PARAMETER SUMMARY

The control statement parameter summary is always generated. An example of the report's format and content is illustrated in figure 4-2.

REPORT 1. MESSAGE LOAD SUMMARY PER TEST

The message load summary per test is optional. It is selected by the R=1 parameter option on the REPORTR call statement. The character count includes the network block header and the message text characters. An example of the format and content of report 1 is illustrated in figure 4-3.

REPORT 2. MESSAGE LOAD SUMMARY FOR TERMINAL xxxxxxx

The terminal message load summary report is optional. It is selected by the R=2 parameter option on the REPORTR call statement and it produces one report per terminal. The character count includes the network block header and the message text characters. An example of the report's format and content is illustrated in figure 4-4.

```
REPORTR - STATISTICAL REPORT GENERATOR

REPORTR CONTROL CARD PARAMETER SUMMARY

INPUT LOG FILE NAME (LF)                          LOGNPS
DUMP START TIME IN SECONDS                             0
DUMP END TIME IN SECONDS                           99999
DUMP NETWORK CONFIGURATION MESSAGES                YES
TIME SLICE START TIME IN SECONDS                       0
TIME SLICE END TIME IN SECONDS                     99999

STIMULATOR NETWORK CONFIGURATION INFORMATION
DATE  79/01/08. TIME  15.02.36.
STIMULATOR 79/01/08. 14.51.23.STIM(LC=LCF,NC=NCF)
LNODE(30)
IS(1/10)
TERM(LN=LINE304,TN=TM304A,SN=YYY,LS=110)
```

Figure 4-2. Example of Default NPS Statistical Report

```
REPORTR - STATISTICAL REPORT GENERATOR

          NPS STATISTICAL REPORT NUMBER 1

          MESSAGE LOAD SUMMARY PER TEST
          -----------------------------

TOTAL ELAPSED TIME -                                  498
TOTAL MESSAGES UPLINE -                                18
AVERAGE MESSAGE LENGTH UPLINE -                       112
TOTAL DATA CHARACTERS UPLINE -                       1435
TOTAL CHARACTER COUNT UPLINE -                       2023
AVERAGE NUMBER OF CHARACTERS PER SECOND UPLINE -        4
TOTAL MESSAGE DOWNLINE -                               28
AVERAGE MESSAGE LENGTH DOWNLINE -                      91
TOTAL DATA CHARACTERS DOWNLINE -                     2041
TOTAL CHARACTER COUNT DOWNLINE -                     2566
AVERAGE NUMBER OF CHARACTERS PER SECOND DOWNLINE -      5
TOTAL DATA CHARACTERS BROADCAST -                       0
TOTAL CONTROL MESSAGES -                              115
TOTAL LOGFILE MESSAGES -                                5
TOTAL NETWORK SERVICE MESSAGE -                        46
```

Figure 4-3. Example of NPS Statistical Report Number 1

```
REPORTR - STATISTICAL REPORT GENERATOR

          NPS STATISTICAL REPORT NUMBER 2

          MESSAGE LOAD SUMMARY FOR TERMINAL XXXXXXXX


TERMINAL ELAPSED TIME -                               484
TOTAL MESSAGES UPLINE -                                18
AVERAGE MESSAGE LENGTH UPLINE -                        93
TOTAL DATA CHARACTERS UPLINE -                       1435
TOTAL CHARACTER COUNT UPLINE -                       1691
AVERAGE NUMBER OF CHARACTERS PER SECOND UPLINE -        3
TOTAL MESSAGE DOWNLINE -                               28
AVERAGE MESSAGE LENGTH DOWNLINE -                      82
TOTAL DATA CHARACTERS DOWNLINE -                     2041
TOTAL CHARACTER COUNT DOWNLINE -                     2301
AVERAGE NUMBER OF CHARACTERS PER SECOND DOWNLINE -      4
```

Figure 4-4. Example of NPS Statistical Report Number 2

REPORT 3. RESPONSE TIME FREQUENCY FOR TERMINAL xxxxxxx

The report of response time frequency for a terminal is optional. It is selected by the R=3 parameter option on the REPORTR call statement and it produces one report for a terminal.

The report presents the interval response time frequency distribution calculation for each script. The format and content of the report are illustrated in figure 4-5.

REPORT 4. RESPONSE TIME FREQUENCY FOR TERMINAL xxxxxxx BY CLASS

The report of response time frequency for a terminal by class is optional. It is selected by the R=4 parameter option on the REPORTR call statement.

The report presents the interval response time frequency distribution calculations for script statement classes. If this report is selected, each terminal (script) using class statements results in a generated report. The format and content of the report are illustrated in figure 4-6.

REPORT 5. RESPONSE TIME FREQUENCY FOR ALL TRANSACTIONS BY CLASS

The report of response time frequency for all transactions by class is optional. It is selected by the R=5 parameter option on the REPORTR call statement. The number of reports produced is dependent on the defined number of classes (class of zero included).

The report presents the interval response time frequency distribution calculations for all classes during the entire test. The content and format of the report are illustrated in figure 4-7.

REPORT 6. RESPONSE TIME FREQUENCY FOR ALL TRANSACTIONS BY ACTIVE TERMINAL SUBSET

The report of response time frequency for all transactions by active terminal subset is optional. It is selected by the R=6 parameter option on the REPORTR call statement. The number of reports generated depends on the number of subsets for the stimulator run.

The report presents the interval response time frequency distribution calculations for each active terminal subset. The format and content of the report are illustrated in figure 4-8.

REPORT 7. SUMMARY OF RESPONSE TIME FOR ALL TRANSACTIONS

The report of summary response times is optional. It is selected by the R=7 parameter option on the REPORTR call statement and it produces one report per terminal.

The report presents the response times for all transactions for the entire test. The report also generates the average response times by active terminal subset occurring during the entire test. The format and content of the report are illustrated in figure 4-9.

ERRORS AND TERMINATION

The REPORTR program validates the parameters specified on the call statement, and terminates if the parameters or parameter values are illegal. It will also terminate with an error message if the logfile is empty, or if the logfile does not contain STIMs zero-time record. The REPORTR program issues a message to the job dayfile upon normal completion of the program.

REPORTR - STATISTICAL REPORT GENERATOR

NPS STATISTICAL REPORT NUMBER 3

RESPONSE TIME FREQUENCY FOR TERMINAL XXXXXXX

TERMINAL ELAPSED TIME =	484
MINIMUM RESPONSE TIME =	0.50
TOTAL RESPONSE TIME =	87
AVERAGE RESPONSE TIME =	4.88
MAXIMUM RESPONSE TIME =	15.78
NUMBER OF TRANSACTIONS =	18
STANDARD DEVIATION =	5.56

INTERVAL SEC	FREQUENCY NO.	CUMULATIVE FREQUENCY	PERCENT FREQUENCY	CUMULATIVE % FREQUENCY
0-2	12	12	66.66	66.66
2-5	1	13	5.55	72.22
5-10	1	14	5.55	77.77
10-15	1	15	5.55	83.33
15-20	3	18	16.66	99.99
20-30	0	18	0.00	99.99
30-40	0	18	0.00	99.99
40-50	0	18	0.00	99.99
50---	0	18	0.00	99.99

Figure 4-5. Example of NPS Statistical Report Number 3

```
REPORTR - STATISTICAL REPORT GENERATOR

          NPS STATISTICAL REPORT NUMBER 4

          RESPONSE TIME FREQUENCY FOR TERMINAL XXXXXXX
                    BY CLASS


CLASS =                                                    0


TERMINAL ELAPSED TIME =                                  484
MINIMUM RESPONSE TIME =                                 0.50
TOTAL RESPONSE TIME =                                     87
AVERAGE RESPONSE TIME =                                 4.88
MAXIMUM RESPONSE TIME =                                15.78
NUMBER OF TRANSACTIONS =                                  18
STANDARD DEVIATION =                                    5.56

INTERVAL    FREQUENCY    CUMULATIVE    PERCENT      CUMULATIVE
  SEC         NO.        FREQUENCY     FREQUENCY    % FREQUENCY
--------    ---------    ----------    ---------    -----------
   0-2          12           12          66.66         66.66
   2-5           1           13           5.55         72.22
   5-10          1           14           5.55         77.77
  10-15          1           15           5.55         83.33
  15-20          3           18          16.66         99.99
  20-30          0           18           0.00         99.99
  30-40          0           18           0.00         99.99
  40-50          0           18           0.00         99.99
  50---          0           18           0.00         99.99
```

Figure 4-6. Example of NPS Statistical Report Number 4

```
REPORTR - STATISTICAL REPORT GENERATOR

          NPS STATISTICAL REPORT NUMBER 5

          RESPONSE TIME FREQUENCY FOR ALL TRANSACTIONS
                    BY CLASS


CLASS =                                                    0


TOTAL ELAPSED TIME =                                     498
MINIMUM RESPONSE TIME =                                 0.50
TOTAL RESPONSE TIME =                                     87
AVERAGE RESPONSE TIME =                                 4.88
MAXIMUM RESPONSE TIME =                                15.78
NUMBER OF TRANSACTIONS =                                  18
STANDARD DEVIATION =                                    5.56

INTERVAL    FREQUENCY    CUMULATIVE    PERCENT      CUMULATIVE
  SEC         NO.        FREQUENCY     FREQUENCY    % FREQUENCY
--------    ---------    ----------    ---------    -----------
   0-2          12           12          66.66         66.66
   2-5           1           13           5.55         72.22
   5-10          1           14           5.55         77.77
  10-15          1           15           5.55         83.33
  15-20          3           18          16.66         99.99
  20-30          0           18           0.00         99.99
  30-40          0           18           0.00         99.99
  40-50          0           18           0.00         99.99
  50---          0           18           0.00         99.99
```

Figure 4-7. Example of NPS Statistical Report Number 5

```
REPORTR - STATISTICAL REPORT GENERATOR

          NPS STATISTICAL REPORT NUMBER 6

          RESPONSE TIME FREQUENCY FOR ALL TRANSACTIONS
                    BY ACTIVE TERMINAL SUBSET


ACTIVE TERMINAL SUBSET =                                        1


SUBSET ELAPSED TIME =                                         464
MINIMUM RESPONSE TIME =                                      0.50
TOTAL RESPONSE TIME =                                          87
AVERAGE RESPONSE TIME =                                      4.88
MAXIMUM RESPONSE TIME =                                     15.78
NUMBER OF TRANSACTIONS =                                       18
STANDARD DEVIATION =                                         5.56

INTERVAL    FREQUENCY    CUMULATIVE    PERCENT      CUMULATIVE
  SEC         NO.        FREQUENCY     FREQUENCY    % FREQUENCY
--------    ---------    ----------    ---------    -----------
   0-2          12           12         66.66         66.66
   2-5           1           13          5.55         72.22
   5-10          1           14          5.55         77.77
  10-15          1           15          5.55         83.33
  15-20          3           18         16.66         99.99
  20-30          0           18          0.00         99.99
  30-40          0           18          0.00         99.99
  40-50          0           18          0.00         99.99
  50---          0           18          0.00         99.99
```

Figure 4-8. Example of NPS Statistical Report Number 6

```
REPORTR - STATISTICAL REPORT GENERATOR

          NPS STATISTICAL REPORT NUMBER 7

          RESPONSE TIME FOR ALL TRANSACTIONS


NUMBER OF TERMINALS =                                           1


TOTAL ELAPSED TIME =                                          498
MINIMUM RESPONSE TIME =                                      0.50
TOTAL RESPONSE TIME =                                          87
AVERAGE RESPONSE TIME =                                      4.88
MAXIMUM RESPONSE TIME =                                     15.78
NUMBER OF TRANSACTIONS =                                       18
STANDARD DEVIATION =                                         5.56


          AVERAGE RESPONSE TIME BY ACTIVE TERMINAL SUBSET

            ACTIVE              AVERAGE
           TERMINAL          RESPONSE TIME
            SUBSET           SEC  -  MSEC
            ------           ------------
              1                4      882
```

Figure 4-9. Example of NPS Statistical Report Number 7

The script writer must pay special attention to the type of terminal for which the script is being written. Basic differences exist between the three terminal types that can be simulated using the Network Products Stimulator. These differences impose certain restrictions on the manner in which the script should be written, including the specification of the terminal type in the SCRIPT directive. Actual script examples appear later in this section.

MODE4 TERMINAL SCRIPT WRITING

The MODE4 terminal is represented by only one composite script. This script is used to define and control the workload for display, card reader, and line printer functions for a 200 User Terminal. Because MODE4 terminals allow both batch and interactive operations, it is important that the desired mode of operation be specified in the script to synchronize the mode of the script with the mode of the device currently being simulated. The mode of the script is explicitly declared using the mode directive:

$$=\text{MODE}\begin{Bmatrix}\text{INTERACTIVE}\\ \text{BATCH}\end{Bmatrix}$$

The initial mode of the script is assumed to be interactive.

A combination of LNODE or RNODE, and TERM input statements is required in the STIM job deck to define the parameters necessary for MODE4 terminal simulation. Only one TERM input statement is necessary to define the console, card reader, and line printer of a MODE4 (200 UT) device. This statement must correspond to the Network Definition Language LCF TERMINAL statement of the console. NPS cannot simulate a MODE4 terminal with the priority flag set on the NDL terminal statement.

ASYNCHRONOUS TERMINAL SCRIPT WRITING

The asynchronous terminal is represented by one composite script written to define and control the workload for all terminal input and output correspondence with the host system. Asynchronous terminals operate only in interactive mode; therefore, no mode directive is necessary in a script to be used solely for this terminal type. The script remains in interactive mode.

If a MODE directive is encountered in a script declared ASYNC, a nonfatal error results during script compilation; a BSEND or BMESSAGE directive causes a fatal error. The LNODE, RNODE, and TERM input statements are required in the STIM job deck to define parameters necessary for asynchronous terminal simulation.

HASP STATION SCRIPT WRITING

Scripts for HASP (Houston Automatic Spooling Program) type stations differ from the previous terminal scripts described. The workload for a HASP station can be logically separated into three functional processes: batch input, batch output, and interactive input/output. The simulation of these three processes requires three separate scripts:

1. The console display script defines only the interactive correspondence to control the HASP station. Batch input and output is represented by the other scripts. A TERM input statement must be used to define the console display and its associated script.

2. Card reader input scripts define the batch input for a HASP station card reader. Interactive control of card reader scripts is performed by the associated interactive console script. The input necessary to coordinate parameters for the card reader and its associated script is supplied by the HASPCR input statement read by STIM.

3. Batch output scripts define the verification process for output files received from the host system by the HASP station and subsequently routed to some output device. Interactive control of batch output scripts is performed by the associated interactive console script. The input necessary to coordinate parameters for output file processing and the associated script is supplied by the HASPLP input statement read by STIM.

JOB AND FILE IDENTIFICATION PROCEDURE

The functional operation of the HASP workstation makes it difficult for the user to predict when output is forthcoming, and to what batch device output from the host system will be routed. For example, any available line printer can receive an output file for listing. Furthermore, each output file can require unique verification processing and, therefore, requires identification.

The association of a particular output stream to a connection is under control of the network application program. The script writer must be aware of the criteria used by the application so that the script receives the correct data.

For users of the Remote Batch Facility (RBF), the association of a particular queue file to a connection can be controlled by using the forms control code capability. When the operating system ROUTE control statement and the RBF SET command are used in combination, RBF prints a file requiring a particular forms control code only on a printer with that forms control code assigned.

The user can also assign a single script (sidx parameter) to all line printers, and then within that script, test and branch to the section that contains the appropriate procedures for the data coming downline. The test on the data can only be made by comparing its length to a known line count. The IF MATCH directive cannot be used with batch data.

JOB DECK STRUCTURE

In addition to the LNODE, RNODE, and TERM input statements, the aforementioned HASPCR and HASPLP input statements are required in the STIM job deck to define necessary parameters for HASP station simulation. Only one TERM input statement is allowed per station.

Two examples of input statements are shown in figure 5-1. The first set of statements (example 1) defines the parameters to simulate the console, three card readers, and three line printers for one HASP station. The second set of statements (example 2) defines the parameters to simulate two HASP stations.

```
Example 1

    LNODE(2)
    TERM(LN=LINES,TN=CRT,SN=CCC,UN=12345)
    HASPCR(ABC,XYZ,DEF)
    HASPLP(XXX=ABCD,YYY=EFGH,ZZZ=IJKL)

Example 2

    LNODE(4)
    TERM(LN=LINE2,TN=CRT,SN=CCC,UN=12345)
    HASPCR(ABC,DEF,GHI,JKL,MNO,PQR,STU)
    HASPLP(XXX=AAAA,YYY=BBBB,ZZZ=EEEE)
    HASPLP(
    .
    .
    .
    HASPLP(
    TERM(LN=LINE4,TN=DIS,SN=ODD,UN=123)
    HASPCR(AAA,BBB)
    HASPLP(PPP=KKKK)
```

Figure 5-1. HASP Deck Structure Examples

In example 1, the console display, CRT, identified by the TERM input statement, is to be simulated by script CCC. Three card reader devices are to be simulated by scripts ABC, XYZ, and DEF. Three line printer devices (ABCD, EFGH, and IJKL) are to be simulated and their output processed by scripts XXX, YYY, and ZZZ. The station is connected to local NPU node 2.

In example 2, the console display, CRT, identified by the first TERM input statement, is to run script CCC. Seven card readers identified by the HASPCR input statement are to be simulated. The HASPLP input statements identify the output processing scripts and associated terminals. The second TERM input statement identifies a second HASP station console display, DIS. The remaining statements identify two card readers and the output processing script and terminal for this second station.

HASP SCRIPT RESTRICTIONS

The initial and only mode of the console display script is interactive. The initial and only mode of the card reader input script and batch output script is batch.

SCRIPT EXAMPLES

The following script examples were selected to illustrate different methods of login and logout procedures and to highlight certain basic aspects of script writing. The larger COBOL and BASIC programs which were contained within the scripts have been removed to conserve space. These scripts should function successfully with the insertion of any comparable program.

As shown, scripts can be created by including all of the script messages individually in the script where needed, or grouped together at one point in the script, or called from another source.

All scripts must be compiled by the SCRIPT compiler before submitting them to STIM. The errors flagged by the SCRIPT compiler must be corrected before STIM can be executed.

User-written scripts can be compiled by supplying a minimal number of job statements as illustrated below:

```
TEST,T777.
USER(username)
SCRIPT.
7/8/9 (multipunched in column 1)

    User-written script here

7/8/9 (multipunched in column 1)
6/7/8/9 (multipunched in column 1)
```

Note the use of the ≤H end-of-line character in the IMESSAGEs and the ISENDs to simulate the end of that line. The ≤H is equivalent to pressing the SEND key at a terminal. When an ≤H or ≤I appears in the text to signify carriage return or line feed, nothing else on that physical line is considered part of the text. For example, if the line ABC≤HDEF≤H follows an =ISEND, only ABC will be sent upline as an MSG block. If the line ABC≤IDEF≤H follows an =ISEND, only ABC will be sent upline as a BLK block.

MODE4 SCRIPT EXAMPLE

The following MODE4 script, shown in figure 5-2, tests the RBF file printing capability for one or more files. The first downline message expected is the FAMILY: prompt. The =IF MATCH directive is used to ensure that the FAMILY prompt is received before sending the login procedure upline. If the first downline message is not the FAMILY prompt, an error in login is logged, and the script is exited.

When the FAMILY prompt is encountered, the SIGNON message furnishes the abbreviated login procedure. Two wait directives ensure that the prompt responses in the SIGNON message are acknowledged before proceeding. The PURALL message clears the queues of any data which might not have been purged by a previous user.

The GOCR message turns on the card reader; the following wait message directive ensures that the card reader is actually on before proceeding. The R message can be used to disable the normal wait following a GOCR (20 seconds) because the message has been received acknowledging that the card reader is ready.

```
=SCRIPT     RAE  MODE4
=COMMENT ** RAE TESTS FILE PRINTING
=COMMENT ** CAPABILITY OF ONE
=COMMENT ** OR MORE FILES
=COMMENT * CASE 1=ONE FILE PRINTING
=COMMENT * CASE 2=MULTIPLE FILE PRINTING
=COMMENT * UNIT-LOGON ----------------
=LOG   LOGIN REQUEST ISSUED
=IF MATCH LINE  JUMP TO LOGIN00
 FAMILY
=LOG ERROR IN LOGIN
=EXIT
=LABEL   LOGIN00
=ISEND SIGNON
=WAIT
=WAIT
=LABEL LOGIN04
=LOG  END LOGON PROCEDURE
=ISEND PURALL
=WAIT
=WAIT
=COMMENT * END OF UNIT-LOGON ----------
=ISEND        GOCR
=WAIT MSG
=ISEND        R
=MODE BATCH
=COMMENT * CASE 1=≠GO LP≠ WITH ONE FILE
=BSEND DRBF10
=MODE INTERACTIVE
=ISEND        GOLP
=WAIT MSG
=ISEND           R
=MODE BATCH
=WAIT  MSG
=* VERIFY THAT DRBF10 OUTPUT IS PRINTED
=LOG      END CASE 1
=* CASE 2=GO LP WITH MULTIPLE PRINT FILES
=MODE INTERACTIVE
=ISEND       GOCR
=WAIT MSG
=ISEND          R
=MODE BATCH
=BSEND DRBF10A
=LABEL ENTRY03
=WAIT MSG
=REPEAT ENTRY03 2 TIMES
=COMMENT: VERIFY THAT 3 FILES ARE PRINTED
=LOG SCRIPT ENDS RAE
=COMMENT * UNIT-LOGOFF ----------------
=MODE INTERACTIVE
=ISEND         END
=IF MATCH WITH RBFEND JUMP TO ENTRY10
=LOG  RBF END PROCEDURE BAD
=EXIT
=LABEL ENTRY10
=IF MATCH WITH APPDESI JUMP TO LOGOFF2
=LOG ERROR IN END PROCEDURE BAD
=EXIT
=IMESSAGE SIGNON
•NPIE274,,RBF≤H
=IMESSAGE PURALL
PURGE ALL≤H
=IMESSAGE GOCR
GO CR≤H
=IMESSAGE LOGOUT
LOGOUT≤H
=IMESSAGE R
```

Figure 5-2. MODE4 Script Example (Sheet 1 of 2)

```
R≤H
=CMESSAGE RBFEND
RBF ENDED
=CMESSAGE RESTART
TO RESTART
=IMESSAGE END
END≤H
=IMESSAGE GOLP
GO LP≤H
=CMESSAGE APPDESI
APPLICATION
=LABEL LOGOFF2
=ISEND          LOGOUT
=LABEL XXLOGO
=IF MATCH WITH RESTART JUMP TO ENTEXIT
=REPEAT XXLOGO 5 TIMES
=LOG  ERROR IN LOGOUT PROCEDURE
=EXIT
=LABEL ENTEXIT
=EXIT
=COMMENT * END OF UNIT-LOGOUT ----------
=BMESSAGE DRBF10
         } USER SUPPLIED COBOL PRINT TEST
=BMESSAGE DRBF10A
         } USER SUPPLIED COBOL PRINT TEST
=ENDSCRIPT
```

Figure 5-2. MODE4 Script Example (Sheet 2 of 2)

The MODE BATCH directive is necessary because a later =WAIT MSG in batch mode will cause execution to wait until batch output from the upline job is received. If this MODE BATCH directive is omitted, the script will be out of synchronization - and no batch message will come downline to satisfy the =WAIT MSG directive.

The =BSEND messages DRBF10 and DRBF10A furnish the COBOL programs (omitted) to test the single and multiple file printing capabilities.

The MODE INTERACTIVE directive returns the script to interactive mode to turn on the line printer with the GOLP message, after which the mode is returned to batch for receiving the output from the COBOL program.

The test process is then repeated using the DRBF10A program to test multiple file printing. A successful test ends through the normal sequence of END, LOGOUT, EXIT messages. The END exits RBF, the LOGOUT exits NVF, and the EXIT script directive informs STIM that the script is finished.

ASYNC SCRIPT EXAMPLES

The first ASYNC script, shown in figure 5-3, illustrates the use of an Interactive Facility (IAF) subsystem for running a BASIC program. The BASIC program can be any program the user chooses. Much of the program included in this example has been omitted to conserve space. Each IAF statement and each BASIC statement is preceded by an =ISEND directive and followed by a =WAIT MSG directive except when no response is expected. Each interactive message is sent upline and a downline response is usually awaited to acknowledge the message.

```
=SCRIPT,336,ASYNC
=COMMENT - SCRIPT USES IAF
=COMMENT - FOR BASIC PROGRAM
=LOGGING IS ON
=LOG LOGIN IN PROGRESS
=IF MATCH LINE JUMP TO STP02
 FAMILY:
=LOG LOGIN FAILED
=EXIT
=LABEL STP02
=ISEND
IAFFAM≤H
=WAIT MSG
=ISEND
USER,PSWD≤H
=WAIT MSG
=ISEND
IAF≤H
=WAIT MSG
=ISEND
CHARGE(XXXX,XXXXXX)≤H
=WAIT MSG
=ISEND
BASIC≤H
=WAIT MSG
=ISEND
OLD,IDA≤H
=WAIT MSG
=ISEND
REWIND,IDA≤H
=WAIT MSG
=ISEND
RNH≤H
=WAIT MSG
=ISEND
LIS,2240≤H
=WAIT MSG
=ISEND
2260 GO TO 1040
=*  NO =WAIT MSG IS NEEDED HERE AS
=*  NO RESPONSE IS EXPECTED
=ISEND
REWIND,IDA≤H
=WAIT MSG
=ISEND
RNH≤H
=WAIT MSG
 }RUNNING OF BASIC PROGRAM CONTINUES HERE
=ISEND
DAYFILE≤H
=WAIT MSG
=ISEND
BYE≤H
=WAIT MSG
=LOG LOGOUT IN PROGRESS
=EXIT
=ENDSCRIPT
```

Figure 5-3. ASYNC Script Example 1

The first downline message expected is the FAMILY: prompt. If the first downline message is not the FAMILY prompt, an error in login is logged, and the script is exited. When the FAMILY prompt is received, the family name prompt response IAFFAM is sent upline. The remaining login prompts are satisfied with the message supplying the username and password.

The application is selected by the IAF message. The subsequent CHARGE message supplies the necessary accounting information required by the installation. Then, the subsystem BASIC is selected by the next IAF message.

Following the BASIC program and the DAYFILE message, BYE is entered to exit IAF, before EXIT causes STIM to terminate this script.

The second ASYNC script, shown in figure 5-4, illustrates a very simple script.

```
=SCRIPT,EX,ASYNC
=WAIT MSG
=WAIT GLOBAL MSG
=ISEND
,RAC0133,,TVF≤H
1≤H
LINE1 TO LOOPBACK≤H
LINE2 TO LOOPBACK≤H
ENDL≤H
END≤H
BYE≤H
=EXIT
=ENDSCRIPT
```

Figure 5-4. ASYNC Script Example 2

The Terminal Verification Facility, TVF, provides a loopback test to echo back the text provided. The test is specified by the number 1.

The =WAIT MSG causes STIM to wait for a downline MSG block, which should include the FAMILY: prompt. The =WAIT GLOBAL MSG sets up the global wait procedure. Each time a block is sent upline to the application, STIM will wait for an MSG block response before continuing script execution.

In this example, when the FAMILY: prompt is received, the abbreviated login response is sent upline. When the message is received acknowledging the login, the message 1 is sent to select the loopback test. When the MSG block response is received, the text LINE1 TO LOOPBACK, LINE2 TO LOOPBACK is echoed back to the terminal.

The ENDL exits the loopback test, END exits TVF, BYE exits NVF, and =EXIT causes STIM to terminate this script.

The third ASYNC script example, shown in figure 5-5, illustrates a more complex abbreviated login procedure and uses the LE (less than or equal) line count verification.

When the message 178/ (the first four characters of the banner) is received downline, the login procedure can begin. If 178/ is not received, the script is exited after logging that an error exists.

```
=SCRIPT PAH ASYNC
=* USES ABBREVIATED LOGIN AND LINE COUNT
=* VERIFICATION
=* CASE 1: ABBREVIATED LOGIN USING
=* MESSAGE STATEMENT
=IF MATCH WITH FAMN JUMP TO RESP
=LOG ERROR IN LOGON  FAMILY
=EXIT
=LABEL RESP
=ISEND LOGIN LOG
=LABEL LOG
=IF MATCH WITH TAPPLN JUMP TO LOGEND
=LOG ERROR IN LOGIN
=COMMENT - TRY LOGIN AGAIN
=WAIT MSG
=ISEND LOGMSG
=GO TO LOG
=LABEL LOGEND
=LOG CASE 1 COMPLETED
=* CASE 2: LINE COUNT VERIFICATION - LE
=IF MATCH WITH ENTCOM JUMP TO LINES
=LOG ERROR IN INITIALIZATION
=EXIT
=LABEL LINES
=ISEND TST03
=WAIT MSG
=IF LINES LE 12 JUMP TO LOOPED
=LOG LINES MORE THAN REQUIRED
=EXIT
=LABEL LOOPED
=LOG CASE 2 COMPLETED
=WAIT MSG
=IF MATCH WITH TOEND JUMP TO OUT4
=LOG ≠TO END≠ NOT FOUND
=EXIT
=LABEL OUT4
=ISEND END*
=WAIT MSG
=IF MATCH WITH ENTCOM JUMP TO OUT1
=LOG ERROR END PROCEDURE1
=LABEL OUT1
=ISEND
TST04 $H
=LOG TST04 SENT
=WAIT MSG
=ISEND
BYE$H
=WAIT MSG
=EXIT
=CMESSAGE FAMN
178/
=IMESSAGE LOG
,≠≠≠≠≠≠≠,,TAPPL4 $H
=CMESSAGE TAPPLN
 TM
=IMESSAGE LOGMSG
,NPIE274,,TAPPL4 $H
=CMESSAGE ENTCOM
ENTER COMMAND
=IMESSAGE TST03
N403 $H
=IMESSAGE END*
END $H
=CMESSAGE TOEND
**ENTER *END TO END TEST
=ENDSCRIPT
```

Figure 5-5. ASYNC Script Example 3

If the first four characters 178/ are received, the abbreviated login procedure is supplied by the message LOG. The seven ≠ characters are replaced with the user name from the TERM statement before the message is sent upline. The first response to this login procedure is expected to start with the characters TM which, in this example, identify the terminal. If the first characters are not TM, an error is logged and login is attempted using the alternate abbreviated login procedure defined in message LOGMSG. The characters TM are again checked for until they are received, at which point login completes. Each unsuccessful match attempt causes an error to be logged.

Once login completes successfully, LE is used to check the application. When the ENTER COMMAND downline prompt is received, the message N403 is sent upline. When this message is acknowledged (WAIT MSG) the number of lines in the second message received is checked to ensure that it is not more than the 12 lines expected. If more than 12 lines are received, the script is exited with an error. If twelve or fewer lines are received, a termination procedure which approximates a user ending the session is shown. TAPPL4 is a test application; N403 is one of the tests contained in TAPPL4 that generates the 12 lines of text tested for above.

HASP SCRIPT EXAMPLE

The following HASP example, shown in figure 5-6, illustrates three interdependent scripts. Script TA1 is the console script which defines the interactive correspondence of the HASP station, script TA2 is a card reader input script, and script TA3 is a line printer batch output script. Script TA1 executes until an event wait causes it to suspend execution; TA1 then waits until that event is turned on in another script.

Only script TA1 need log in because it simulates the console device of the HASP terminal. The first downline message expected is the FAMILY: prompt. If this prompt is not received, the script exits and an error message is logged. The FAMILY prompt response supplies the user name and the application name RBF in the abbreviated login format. The following two wait message directives ensure that login is successful before continuing to the purge statement which clears the queues of any data which might have been left in the queues by a previous user.

The line printer forms code is supplied so that output with a forms code of FF will be printed on line printer LP1. Then the card reader is turned on. At this point, the event LOGDIN is turned on, and execution of script TA2 can proceed. TA2 sends one batch job upline. The ROUTE statement causes the output from that job to be printed only at a line printer having the forms code FF. Script TA1 waits for event CR1RDY, which signals the end of the card reader script TA2. The interactive message GO LP1 turns on the line printer; however, the line printer script TA3 is waiting for event LP1GO to be turned on which is done by the next directive in script TA1. Output from the job sent upline by script TA2 is printed until at least 30 lines have been printed, at which point it waits for the rest of the mesage, sets LP1RDY on, and exits. Then script TA1 enters its termination sequence by sending END to exit from RBF, BYE to exit from NVF, and EXIT to inform STIM that the HASP script is completed.

```
=SCRIPT,TA1,HASP
=LOGGING IS ON
=IF MATCH LINE JUMP TO STP0002
 FAMILY:
=LOG LOGIN FAILED
=EXIT
=LABEL STP0002
=ISEND
,RAC0001,,RBF$H
=WAIT MSG
=WAIT MSG
=ISEND
PURGE ALL$H
=WAIT MSG
=ISEND
SET LP1 FMS=FF$H
=WAIT MSG
=ISEND
GO CR1$H
=WAIT MSG
=LOG EVENT LOGDIN IS ON
=EVENT ON LOGDIN
=EVENT WAIT CR1RDY
=ISEND
GO LP1$H
=WAIT MSG
=LOG LP1GO EVENT ON
=EVENT ON LP1GO
=EVENT WAIT LP1RDY
=ISEND
END$H
=WAIT MSG
=WAIT MSG
=ISEND
BYE$H
=WAIT MSG
=LOG LOGGING OFF NOW
=EXIT
=ENDSCRIPT


=SCRIPT,TA2,HASP
=EVENT WAIT LOGDIN
=BSEND
JOB.
USER(RJP0030)
BANNER.   03A
ROUTE(OUTPUT,DEF,DC=PR,FC=FF)
COPY,INPUT,OUTPUT.
=EOR
LINE1
LINE2
LINE3
=EOI
=LOG CARD READER SCRIPT COMPLETE
=EVENT ON CR1RDY
=EXIT
=ENDSCRIPT


=SCRIPT,TA3,HASP
=EVENT WAIT LP1GO
=MODE BATCH
=IF LINES GE 30 JUMP TO STLOG
=LOG LINES NOT ENOUGH
=EXIT
=LABEL STLOG
=WAIT MSG
=LOG LINE PRINTER SCRIPT COMPLETE
=EVENT ON LP1RDY
=EXIT
=ENDSCRIPT
```

Figure 5-6. HASP Script Example

STANDARD CHARACTER SET

All message data prepared for input to NPS is represented using the CDC 64-character set which is one of the four character sets represented in table A-1; but, in order to represent and keypunch NPS declared messages using the full ASCII 128-character set, two of the CDC 64-character set characters are designated as escape characters. The two escape characters are the ≤ and the ¬ (internal 6-bit codes 74_8 and 76_8). Thus, two CDC 64-character set characters are required to represent the two characters designated as escape characters and represent all those characters beyond the range of the standard CDC 64-character set. The character set that should be used to prepare message data for use by NPS is shown in table A-2.

The octal ASCII code in table A-1 and table A-2 is shown with no parity.

ASCII 128-CHARACTER SET

The character set employed by NPS is not the same as the standard CDC 64-character set. Table A-3 contains the complete ASCII character set supported by the Network Access Method. The standard Network Operating System 96-character subset consists of the rightmost six columns. Binary designations are shown in table A-3 so that conversion of the leftmost two columns and rightmost two columns to octal or hexadecimal can be made; no octal display code equivalents appear for the characters of these four columns in the preceding conversion tables.

During output operations to a terminal that does not support the full 128- or 96-character sets, conversion from the 96-character set to the 64-character set is accomplished by folding column 6 into column 4 and column 7 into column 5. Converting the 128-character set leaves extra characters. The MODE4 and HASP terminal classes 9 through 15 replace the extra characters with blanks, while terminal classes 1 through 8 replace the extra characters with NULLs.

TABLE A-1. STANDARD CHARACTER SET

Display Code (octal)	CDC			ASCII		
	Graphic	Hollerith Punch (026)	External BCD Code	Graphic Subset	Punch (029)	Code (octal)
00†	: (colon)††	8-2	00	: (colon) ††	8-2	072
01	A	12-1	61	A	12-1	101
02	B	12-2	62	B	12-2	102
03	C	12-3	63	C	12-3	103
04	D	12-4	64	D	12-4	104
05	E	12-5	65	E	12-5	105
06	F	12-6	66	F	12-6	106
07	G	12-7	67	G	12-7	107
10	H	12-8	70	H	12-8	110
11	I	12-9	71	I	12-9	111
12	J	11-1	41	J	11-1	112
13	K	11-2	42	K	11-2	113
14	L	11-3	43	L	11-3	114
15	M	11-4	44	M	11-4	115
16	N	11-5	45	N	11-5	116
17	O	11-6	46	O	11-6	117
20	P	11-7	47	P	11-7	120
21	Q	11-8	50	Q	11-8	121
22	R	11-9	51	R	11-9	122
23	S	0-2	22	S	0-2	123
24	T	0-3	23	T	0-3	124
25	U	0-4	24	U	0-4	125
26	V	0-5	25	V	0-5	126
27	W	0-6	26	W	0-6	127
30	X	0-7	27	X	0-7	130
31	Y	0-8	30	Y	0-8	131
32	Z	0-9	31	Z	0-9	132
33	0	0	12	0	0	060
34	1	1	01	1	1	061
35	2	2	02	2	2	062
36	3	3	03	3	3	063
37	4	4	04	4	4	064
40	5	5	05	5	5	065
41	6	6	06	6	6	066
42	7	7	07	7	7	067
43	8	8	10	8	8	070
44	9	9	11	9	9	071
45	+	12	60	+	12-8-6	053
46	-	11	40	-	11	055
47	*	11-8-4	54	*	11-8-4	052
50	/	0-1	21	/	0-1	057
51	(	0-8-4	34	(	12-8-5	050
52	)	12-8-4	74	)	11-8-5	051
53	$	11-8-3	53	$	11-8-3	044
54	=	8-3	13	=	8-6	075
55	blank	no punch	20	blank	no punch	040
56	, (comma)	0-8-3	33	, (comma)	0-8-3	054
57	. (period)	12-8-3	73	. (period)	12-8-3	056
60	≡	0-8-6	36	#	8-3	043
61	[	8-7	17	[	12-8-2	133
62	]	0-8-2	32	]	11-8-2	135
63	% ††	8-6	16	% ††	0-8-4	045
64	≠	8-4	14	" (quote)	8-7	042
65	→	0-8-5	35	_ (underline)	0-8-5	137
66	∨	11-0 or 11-8-2†††	52	!	12-8-7 or 11-0†††	041
67	∧	0-8-7	37	&	12	046
70	↑	11-8-5	55	' (apostrophe)	8-5	047
71	↓	11-8-6	56	?	0-8-7	077
72	<	12-0 or 12-8-2†††	72	<	12-8-4 or 12-0†††	074
73	>	11-8-7	57	>	0-8-6	076
74	≤	8-5	15	@	8-4	100
75	≥	12-8-5	75	\	0-8-2	134
76	¬	12-8-6	76	~ (circumflex)	11-8-7	136
77	; (semicolon)	12-8-7	77	; (semicolon)	11-8-6	073

†Twelve zero bits at the end of a 60-bit word in a zero byte record are an end of record mark rather than two colons.

††In installations using a 63-graphic set, display code 00 has no associated graphic or card code; display code 63 is the colon (8-2 punch). The % graphic and related card codes do not exist and translations yield a blank (55_8).

†††The alternate Hollerith (026) and ASCII (029) punches are accepted for input only.

TABLE A-2. NPS CHARACTER SET

ASCII Code Standard Print		Display Code		ASCII Code Standard Print		Display Code	
Character	Code Octal	Character	Internal Code 6/12-Bit Octal	Character	Code Octal	Character	Internal Code 6/12-Bit Octal
:	072	:	00	)	051	)	52
A	101	A	01	$	044	$	53
B	102	B	02	=	075	=	54
C	103	C	03	(SP)	040	(SP)	55
D	104	D	04	,	054	,	56
E	105	E	05	.	056	.	57
F	106	F	06	#	043	≡	60
G	107	G	07	[	133	[	61
H	110	H	10	]	135	]	62
I	111	I	11	%	045	%	63
J	112	J	12	"	042	≠	64
K	113	K	13	-	137	→	65
L	114	L	14	!	041	∨	66
M	115	M	15	&	046	∧	67
N	116	N	16	'	047	↑	70
O	117	O	17	?	077	↓	71
P	120	P	20	>	074	<	72
Q	121	Q	21	<	076	>	73
R	122	R	22	@	100	≤ A	7401†
S	123	S	23	\	134	≥	75
T	124	T	24	^	136	≤ B	7402††
U	125	U	25	;	073	;	77
V	126	V	26	`	140	¬ :	7600
W	127	W	27	a	141	¬ A	7601
X	130	X	30	b	142	¬ B	7602
Y	131	Y	31	c	143	¬ C	7603
Z	132	Z	32	d	144	¬ D	7604
0	060	0	33	e	145	¬ E	7605
1	061	1	34	f	146	¬ F	7606
2	062	2	35	g	147	¬ G	7607
3	063	3	36	h	150	¬ H	7610
4	064	4	37	i	151	¬ I	7611
5	065	5	40	j	152	¬ J	7612
6	066	6	41	k	153	¬ K	7613
7	067	7	42	l	154	¬ L	7614
8	070	8	43	m	155	¬ M	7615
9	071	9	44	n	156	¬ N	7616
+	053	+	45	o	157	¬ O	7617
-	055	-	46	p	160	¬ P	7620
*	052	*	47	q	161	¬ Q	7621
/	057	/	50	r	162	¬ R	7622
(	050	(	51	s	163	¬ S	7623
t	164	¬ T	7624	DLE	020	¬ ≡	7660
u	165	¬ U	7625	DC1	021	¬ [	7661
v	166	¬ V	7626	DC2	022	¬]	7662
w	167	¬ W	7627	DC3	023	¬ %	7663
x	170	¬ X	7630	DC4	024	¬ ≠	7664
y	171	¬ Y	7631	NAK	025	¬ →	7665
z	172	¬ Z	7632	SYN	026	¬ ∨	7666
{	173	¬ 0	7633	ETB	027	¬ ∧	7667
;	174	¬ 1	7634	CAN	030	¬ ↑	7670
}	175	¬ 2	7635	EM	031	¬ ↓	7671
~	176	¬ 3	7636	SUB	032	¬ <	7672
DEL	177	¬ 4	7637	ESC	033	¬ >	7673
NUL	000	¬ 5	7640	FS	034	¬ ≤	7674
SOH	001	¬ 6	7641	GS	035	¬ ≥	7675
STX	002	¬ 7	7642	RS	036	¬ ¬	7676
ETX	003	¬ 8	7643	US	037	¬ ;	7677
EOT	004	¬ 9	7644	NULL	---	¬ :	7400
ENQ	005	¬ ;	7645	@	100	¬ A	7401

TABLE A-2. NPS CHARACTER SET (Contd)

ASCII Code Standard Print		Display Code		ASCII Code Standard Print		Display Code	
Character	Code Octal	Character	Internal Code 6/12-Bit Octal	Character	Code Octal	Character	Internal Code 6/12-Bit Octal
ACK	006	¬ -	7646	^	136	≤ B	7402[††]
BEL	007	¬ *	7647	NULL	---	≤ C	7403
BS	010	¬ /	7650	NULL	---	≤ D	7404
HT	011	¬ (	7651	NULL	---	≤ E	7405
LF	012	¬)	7652	NULL	---	≤ F	7406
VT	013	¬ $	7653	NULL	---	≤ G	7407
FF	014	¬ =	7654	---	---	≤ H	7410[†††]
CR	015	¬ (SP)	7655	---	---	≤ I	7411[††††]
SO	016	¬ ,	7656	---	---	≤ J	7412[†††††]
SI	017	¬ .	7657				

[†]The double character ≤ A represents the display character ≤ (@ in ASCII).

[††]The double character ≤ B represents the display character ¬ (^ in ASCII).

[†††]The double character ≤ H represents the end of logical line character used by NPS.

[††††]The double character ≤ I represents the end of physical line character used by NPS.

[†††††]The double character ≤ J represents the cancel character used by NPS.

Note: This character set is represented with no parity. The IVT terminal command PA is not supported by NPS.

TABLE A-3. FULL ASCII CHARACTER SET

128-Character Set

96-Character Subset

64-Character Subset

b_7			0	0	0	0	1	1	1	1
b_6			0	0	1	1	0	0	1	1
b_5			0	1	0	1	0	1	0	1
B_{i_s}										
b_4	b_3	b_2	b_1	ROW / COLUMN →	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	C	FF	FS	,	<	L	\	l	¦
1	1	0	1	D	CR	GS	-	=	M	]	m	}
1	1	1	0	E	SO	RS	.	>	N	^	n	~
1	1	1	1	F	SI	US	/	?	O	_	o	DEL

On output to a terminal that does not support a full 128-character set, the 95- to 64-character set conversion is accomplished by folding column 7 into column 5. Folding the 128- to 95-character subset consists of replacing the excess characters with blanks. The character represented by the hexadecimal digits xy is found at the intersection of column x and row y.

DIAGNOSTIC MESSAGES

Messages that the user might encounter are listed alphabetically in table B-1. Entries in which the first characters might change depending on the parameters of the job in progress are listed near the end. Entries beginning with numbers follow the alphabetic listing.

The format of the diagnostic messages consists of four columns listing the following information:

Message — The message is capitalized with all variables indicated in lowercase letters. Informative messages and diagnostics are included.

Significance — This column briefly describes the problem and defines variables in the message.

Action — This column states the action required and how to perform it.

Issued By — This column states the routine which generates the given message.

TABLE B-1. DIAGNOSTIC MESSAGES

Message	Significance	Action	Issued By
AUTHORIZE NPS xx, xx, xx, xx.	This message flashes the EST ordinals selected for the simulation.	Job may be continued or rolled out for later use.	SIP
BAD LCF/TERM RECORD POSITION	LCF terminal records not found for subsequent configuration of HASP terminals.	Verify LCF, correct problem and rerun.	STIM
BAD OR MISSING NUMERIC FIELD	Fatal error.	Correct error and rerun.	SCRIPT
BAD SCRIPT LIBRARY FILE	SCRIPT library file does not contain EOF.	Correct problem and rerun.	STIM
BADLY NESTED REPEAT	Fatal error.	Correct error and rerun.	SCRIPT
BATCH COMMAND ILLEGAL IN ASYNC SCRIPT	Fatal error.	Correct error and rerun.	SCRIPT
CLASS NUMBER NOT WITHIN LEGAL RANGE	Fatal error.	Correct error and rerun.	SCRIPT
CONFLICTING FORMS OF IS CARD DETECTED	The two different forms of IS input statement have been detected in the same NPS run.	Correct problem and rerun.	STIM
CONTROL CARD ERRORS - RUN ABORTED	Self-explanatory	Correct errors and retry.	SCRIPT
CONTROL TRANSFERRED INTO REPEAT LOOP n	Fatal error. n represents level of nesting.	Correct error and rerun	SCRIPT
COUNTER NAME MUST BE LQ 7 CHARS	Fatal error.	Correct error and rerun.	SCRIPT
DUPLICATE LABEL NAME	Fatal error.	Correct error and rerun.	SCRIPT
DUPLICATE MESSAGE NAME	Fatal error.	Correct error and rerun.	SCRIPT
DUPLICATE SCRIPT NAME	Fatal error.	Correct error and rerun.	SCRIPT
=CHARACTER MISSING	=character missing on input statement.	Correct error and rerun.	STIM

TABLE B-1. DIAGNOSTIC MESSAGES (Contd)

Message	Significance	Action	Issued By
EMPTY INPUT FILE	The local input file specified on STIM call card is empty.	Correct problem and rerun.	STIM
EMPTY NEWSL	The SCRIPT library file specified on the STIM call card is empty.	Correct problem and rerun.	STIM
ENDSCRIPT MISSING AT END OF INPUT	Fatal error.	Correct error and rerun.	SCRIPT
ENDSCRIPT NOT ALLOWED BEFORE SCRIPT	Fatal error.	Correct error and rerun.	SCRIPT
ERRORS IN BINARY - RUN ABORTED	Error in =SCRIPT directive syntax or conflict between entries.	Correct error and rerun.	SCRIPT
ERRORS IN INPUT - RUN ABORTED	Error in =SCRIPT directive keyword.	Correct error and rerun.	SCRIPT
EVENT NAME MUST BE LQ 6 CHARS	Fatal error.	Correct error and rerun.	SCRIPT
FILE NOT FOUND, MUST BE A LOCAL FILE	Error occurs if no local file of the name lfn is found when XSEND FROM lfn is processed.	Check that file lfn is a local file.	SCRIPT
FIRST NUM MUST BE LESS THAN 2ND NUM	Fatal error.	Correct error and rerun.	SCRIPT
FL REQUEST EXCEEDS MAX ALLOWED	Request exceeds central memory indicated on JOB statement. STIM run terminated during initialization.	Correct error and retry.	STIM
FL REQUEST IGNORED	Request for additional central memory during initialization is ignored. STIM will terminate condition continues to exist.	Call system analyst.	STIM
ILLEGAL DEVICE TYPE, DT≠0	Controlling terminal for a HASP card reader is not a console; TERM card does not specify a console.	Correct problem and rerun.	STIM
ILLEGAL FILE NAME	Fatal error.	Correct error and rerun.	SCRIPT
ILLEGAL REASON CODE	Fatal error.	Correct error and rerun.	SCRIPT
ILLEGAL SCRIPT NAME ON LIST	Nonfatal error.	Ignore or correct later.	SCRIPT
ILLEGAL STATEMENT WITHIN A SCRIPT	Fatal error.	Correct error and rerun.	SCRIPT
ILLEGAL TIP TYPE, NOT A HASP	Console specified for a HASP card reader is not specified for a HASP TIP.	Correct specification and rerun.	STIM
INCOMPLETE LCF/TERM. REC READ	Status of incomplete returned during reading an LCF terminal record.	Correct error and retry.	STIM
INFINITE LS FOR TERMINAL xxxxxxx	Informative message to notify the user that the specific terminal is stimulating an infinite output line speed.	None.	STIM

TABLE B-1. DIAGNOSTIC MESSAGES (Contd)

Message	Significance	Action	Issued By
INPUT FILE EMPTY - RUN ABORTED	Self-explanatory.	Correct error and retry.	SCRIPT
INVALID TERMINAL NAME	Simulated HASP terminal name not found in LCF.	Correct problem and rerun.	STIM
LABEL NAME MUST BE LQ 7 CHARS	Fatal error.	Correct error and rerun.	SCRIPT
LCF/TERM NODE MISMATCH	LCF terminal record NPU node ID is not same as that on a LNODE or RNODE input card.	Correct error and retry.	STIM
LCF/TERMINAL xxxxxxx NOT FOUND	LCF terminal record for terminal xxxxxxx was not found.	Correct error and retry.	STIM
LEFT TERMINATOR MISSING	Left terminator is missing on specified input card.	Correct error and retry.	STIM
LEVEL OF REPEAT NESTING MUST BE LE 3	Fatal error.	Correct error and rerun.	SCRIPT
LINE FAILED, TERMINAL xxxxxxx	Caused by =LFAIL directive.	Recovery is automatic.	STIM
LINE RECOVERED, TERM xxxxxxx	Self-explanatory.	Recovered to point where input began.	STIM
LIST STMT WITHIN A SCRIPT	Nonfatal error.	Ignore or correct later.	SCRIPT
LS FOR TERMINAL xxxxxxx DIFFERS FROM LCF VALUE	Informative message for non-auto-speed asynchronous terminals.	None, LS value is honored.	STIM
LS FOR TERMINAL xxxxxxx EXCEEDS AUTO-SPEED MAX	Informative message for auto-speed asynchronous terminals.	None, LS value is honored.	STIM
LLT LINKAGE ERROR	LCF/NCF does not indicate a logical link for a simulated NPU.	Correct problem and rerun.	STIM
MESSAGE NAME MUST BE LQ 7 CHARS	Fatal error.	Correct error and rerun.	SCRIPT
MESSAGE NOT DEFINED OR WRONG TYPE	Fatal error.	Correct error and rerun.	SCRIPT
MODE STMT IN ASYNC SCRIPT	Nonfatal error.	Ignore or correct later.	SCRIPT
MORE THAN 4 xxxx CARDS	Too many LNODE or RNODE input cards.	Correct error and rerun.	STIM
NESTING NOT CHECKED - ERR ON PREV. REPEAT	Fatal error.	Correct error and rerun.	SCRIPT
NODE CARD MISSING	LNODE card is missing in the INPUT file.	Correct problem and rerun.	STIM
NO SYSTEM PRIVILEGES	Application program does not have required system privileges to run simulation program.	Correct problem and rerun. Job must have system origin, which can usually be satisfied by calling STIM from DIS.	SIP
NONFATAL ERRORS IN SCRIPT RUN	Informative message.	Ignore or correct later.	SCRIPT

TABLE B-1. DIAGNOSTIC MESSAGES (Contd)

Message	Significance	Action	Issued By
NUMBER NOT WITHIN LEGAL RANGE	Fatal error.	Correct error and rerun.	SCRIPT
ONLY 6 CTRS ALLOWED PER SCRIPT	Fatal error.	Correct error and rerun.	SCRIPT
OPEN xxx ERROR nn	Initial access to NHP file xxx unsuccessful.	Correct indicated error nn and retry.	STIM
PARAMETER SYMBOL MISSING	Parameter symbol missing on specified input card.	Correct error and retry.	STIM
PARAMETER VALUE MISSING	Parameter value missing on specified input card.	Correct error and retry.	STIM
PASSIVE DEVICE ON TERM CARD	A non-interactive device is specified on a TERM card.	Correct problem and rerun.	STIM
REPEAT LABEL NOT PREVIOUSLY DEFINED	Fatal error.	Correct error and rerun.	SCRIPT
REPORTR - BAD DM PARAMETER NO DUMP ASSUMED	A value other than Y or N specified for DM parameter. N is assumed.	None.	REPORTR
REPORTR - BAD INPUT FILE PROGRAM ABORTED	A wrong input file is given to REPORTR.	Correct problem and retry.	REPORTR
REPORTR - COMPLETED	Successful run.	None.	REPORTR
REPORTR - CONTROL CARD ERROR, PROGRAM ABORTED	Unrecognized keyword in control card.	Correct error and rerun.	REPORTR
REPORTR - DUMP TIME INTERVAL ERROR, PROGRAM ABORTED	D1 is less than D2.	Correct error and rerun.	REPORTR
REPORTR - EMPTY INPUT FILE, PROGRAM ABORTED	Input file not found.	Correct problem and retry.	REPORTR
REPORTR - ILLEGAL NUMBER ON CONTROL CARD	Alphabetic character detected in a field where a numeric value is expected.	Correct problem and retry.	REPORTR
REPORTR - INVALID REPORT NUMBER, PROGRAM ABORTED	Report number greater than 7 or less than 1 specified on control card.	Correct error and rerun.	REPORTR
REPORTR - NO SERVICE MESSAGE TRAFFIC	The STIM run, which created the log file being used, was aborted before initialization.	Correct problem and rerun the STIM and REPORTR runs.	REPORTR
REPORTR - TIME SLICE INTERVAL ERROR, PROGRAM ABORTED	T1 is less than T2.	Correct error and rerun.	REPORTR
REPORTR - ZERO TIME RECORD NOT FOUND	Caused either by a bad STIM run or an incomplete STIM run.	Rerun STIM and/or call analyst.	REPORTR
RIGHT TERMINATOR MISSING	Right terminator missing on specified input card.	Correct error and retry.	STIM
RUN TIME EXCEEDED. NPS TERMINATED	Run time designated on STIM call statement exceeded.	STIM run terminated.	STIM
SCRIPT COMPLETE	SCRIPT compiled successfully. Script library created.	None.	SCRIPT

TABLE B-1. DIAGNOSTIC MESSAGES (Contd)

Message	Significance	Action	Issued By
SCRIPT LIBRARY TOO LONG	Self-explanatory.	Reduce number of scripts and retry.	SCRIPT
SCRIPT NAME MUST BE LQ 3 CHARS	Fatal error.	Correct error and rerun.	SCRIPT
SCRIPT xxx NOT ON SL	Script name xxx not found in script library file specified on STIM call card.	Correct problem and rerun.	STIM
SCRIPT STATEMENT MISSING	Fatal error.	Correct error and rerun.	SCRIPT
SCRIPT -- 7 CHAR MASK NEEDED, ISEND LOGIN	Self-explanatory.	Correct error and rerun.	SCRIPT
SET EST ORDINAL xx ON /CH yy	SIP requests the NP EST ordinal number be turned ON. xx = EST NP ordinal. yy = interface channel.	Set EST ordinal xx on to run this STIM job.	SIP
SIP - ADDRESS NOT IN FL xxxx	SIP addresses CM location not in STIM FL area. xxxx address of SIP calling subroutine.	No operator action required. SIP aborts job.	SIP
SIP - CH/CONFLICT xx -ABORT-	SIP must have a dedicated channel to simulate an NPU. Channel xx conflict in EST table.	Turn off device in EST causing conflict. Rerun.	SIP
SIP - MORE THAN 2-PIPS -ABORT-	SIP only allows two PIPs when simulating more than two NPUs.	No operator action required. SIP aborts job.	SIP
SIP - NO MATCH NPID=xxxx -ABORT-	NPID coupler xxxx not found in EST, or is on in EST.	This NPID coupler must be OFF in the EST. Rerun.	SIP
SIP - NO MSG TERM FROM PIP	SIP received a down line message from PIP with end-of-message terminator missing.	No operator action required. SIP adds end of message terminator and continues with the simulation run.	SIP
SIP - NPU SIMULATION DROP	Control point error flag set. (Word 22 of STIM control point communications area.)	No operator action required. SIP aborts job.	SIP
SIP - Nx BAD FCT/CODE xxxx	Nx = NPU ID number. SIP has received unauthorized function code from PIP. xxxx = illegal function code.	No operator action required. SIP aborts job.	SIP (3NS)
SIP - Nx BAD MSG/LTH STIM xxxx.	Nx = NPU ID number. Character count for upline message from STIM buffer out of range for communication interface message buffer. Character count should be between 3 - 4000. xxxx = wrong character count.	No operator action required. SIP aborts job.	SIP (3NS)
SIP - Nx BAD NPU MEM/ADR xxxx.	Nx = NPU ID number. PIP exceeds maximum memory size, (128K) for the simulated 2550 (NPU). xxxx = upper 8-bits of memory address error.	No operator action required. SIP aborts job.	SIP (3NS)

TABLE B-1. DIAGNOSTIC MESSAGES (Contd)

Message	Significance	Action	Issued By
SIP - Nx BAD PPU ORDER xxxx	Nx = NPU ID number. xxxx = illegal order code. Legal order codes are: 1 = Supervisory service message. 2 = Interactive 3 = Batch 4 = Host to Host 5 = Do nothing (IDLE).	No operator action required. SIP aborts job.	SIP (3NS)
SIP - Nx COUP WRONG FOR NPU STATUS.	Nx = NPU ID number. PIP function code requested SIP to send coupler status to PIP. Coupler status cell in SIP was not correct for a transfer at this time. Channel communication interface error between SIP/PIP.	No operator action required. SIP aborts job.	SIP (3NS)
SIP - Nx CH STILL EMPTY AFT/1SEC.	Nx = NPU ID number. PIP function code requested output. SIP will monitor the channel to go full, then input data from PIP. If channel does not indicate full within one second, this message will occur. Channel communication interface problem between SIP/PIP.	No operator action required. SIP aborts job.	SIP (3NS)
SIP - Nx CH xx ACTIVE AFT/1SEC	SIP sent a block or word of data, or has received a block or word of data to and from PIP. The channel should have been disconnected by PIP within one second after the data transfer has taken place. This did not occur.	No operator action required. SIP aborts job.	SIP (3NS)
SIP - Nx CH xx ACT/EMPTY NO FCT	Nx = NPU ID number; CHxx = interfacing channel. SIP timed out waiting for a function command code from PIP. The channel was active 1 second and empty without a function command code from PIP.	No operator action required. Sip removes the errored NPU, and all others using the same interfacing channel, then check for active NPUs. If none exists, job aborts, otherwise, simulation continues.	SIP (3NS)
SIP - Nx CHxx FULL ON OUTPUT	Nx = NPU ID number; CHxx = interfacing channel. PIP function code requested SIP to output data, status, etc. SIP has found the channel active and also full. Channel communications interface problem between SIP/PIP.	No operator action required. SIP removes the errored channel, and checks for active channels. If none exists, the job aborts, otherwise, simulation continues on the active channels.	SIP (3NS)

TABLE B-1. DIAGNOSTIC MESSAGES (Contd)

Message	Significance	Action	Issued By
SIP - Nx CHxx INACTIVE TO OUT	Nx = NPU ID number; CHxx = channel number. PIPs function code requested SIP to output data, status, etc. SIP will monitor the channel, after the request, until it becomes active, then will do the output. If the channel does not become active within one second, this message will occur.	No operator action required. SIP aborts job.	SIP (3NS)
SIP Nx CHxx IMPROP/CH/ RELEASE	Nx = NPU ID number; CHxx = interfacing channel improper channel release: SIP/PIP out of sync. Channel was not disconnected or released by SIP.	No operator action required. SIP aborts job.	SIP (3NS)
SIP Nx CHxx IS EMPTY TO INP	Nx = NPU ID number; CHxx = interfacing channel. PIP function code requested SIP to receive data. SIP was monitoring the channel to go active, this did not occur within one second. Channel communications interface problem between SIP/PIP.	No operator action required. SIP aborts job.	SIP (3NS)
SIP Nx CHxx NO EQ/NUM xx	Nx = NPU ID number; CHxx = channel number; EQ/NUM xx = equipment number. This message is associated with resident program FUNCT. When a function is received by SIP it is matched with the equipment number for the simulated NPU. If no match for that equipment number, this message will occur.	No operator action required. SIP aborts job.	SIP (3NS)
SIP Nx CHxx NO RESPONSE PIP	Nx = NPU ID number; CHxx = channel number. This message is associated with resident program SETUP. When channel communication is established, then SETUP will start the channel time out cell for this NPU. Resident program FUNCT will clear this cell after each response from PIP. SETUP keeps increasing the cell by one, and when this cell becomes minus, this message will occur. SIP/PIP communication is lost.	No operator action required. SIP aborts job.	SIP (3NS)
SIP Nx INP/BUF ALREADY EMPTY	Nx = NPU ID number. PIP function code requested up-line data. SIP checked STIM up-line buffer area and found it empty. Something is wrong with the channel communications interface between SIP/PIP/STIM.	No operator action required. SIP aborts job.	SIP (3NS)

TABLE B-1. DIAGNOSTIC MESSAGES (Contd)

Message	Significance	Action	Issued By
SIP Nx MXLTH TOO BIG FOR O-BUF	Nx = NPU ID number. PIP function code requested a down-line message. Before SIP requested the message, there was enough buffer space available; but after receiving the message, there isn't enough buffer space for it. Communication interface problem between STIM/SIP/PIP.	No operator action required. SIP aborts job.	SIP (3NS)
SIP - PARAM/LIST BAD -ABORT-	Error in parameter words passed by STIM: a. Fewer than two parameter words. b. More than five parameter words. c. FET buffer pointers to the up-line or down-line messages equal zero.	No operator action required. SIP aborts job.	SIP
SIP STIM ABORTED NO ACT/NPUS	When no NPUs are active, this message will occur. Network has been dropped.	No operator action required. SIP aborts job.	SIP (3NS)
STATEMENT WORD TOO LONG	Fatal error.	Correct error and rerun.	SCRIPT
STIM-EMPTY SCRIPT LIBRARY	No SCRIPT library found.	Check that SCRIPT library has been created.	STIM
STIM - HASP CR OR LP NOT DEFINED IN LCF	HASP scripts are specifying more card readers and/or line printers than are available for that console.	Reduce number of scripts, LCF for card readers and line printers specified.	STIM
STIM - ILLEGAL INPUT CARD, xxxxxx	An illegal input card is encountered after the TERM card is read.	Correct the illegal card of which the first six characters are specified.	STIM
STIM - TOO MANY ACTIVE REPEATS	Abort STIM.	Dump STIM FL.	STIM
STIM INITIALIZING NETWORK	STIM is processing network service messages to configure the simulated network.	None.	STIM
STIM RUN COMPLETE	STIM run has completed all designated work.	STIM run is terminated.	STIM
STIMULATOR RUN STARTED	STIM initialization is complete and terminal to application correspondence has started.	None.	STIM
SYNTAX ERROR	Fatal error.	Correct error and rerun.	SCRIPT
TERM CARD MISSING	TERM card is missing from the input file.	Correct problem and rerun.	STIM
TERM CARD, HASP TRML RANGE ERROR	HASP terminal has range specified; or is in range specified; or occurs before end terminal of range.	Correct error and rerun.	STIM

TABLE B-1. DIAGNOSTIC MESSAGES (Contd)

Message	Significance	Action	Issued By
THE FOLLOWING LABEL WAS NOT FOUND	Fatal error.	Correct error and rerun.	SCRIPT
TOO MANY ACTIVE EVENT NAMES	Number of events in a non-zero state exceeds MAXNEVTS.	Correct error and retry.	STIM
TOO MANY xxxx CHARACTERS	Too many value characters for parameter xxxx on specified input card.	Correct error and retry.	STIM
TOO MANY xxxx DIGITS	Too many value digits for parameter xxxx on specified input card.	Correct error and retry.	STIM
TOO MANY EVENT DECLARATIONS	Fatal error.	Correct error and rerun.	SCRIPT
TOO MANY DOWNLINE BLOCKS OUTSTANDING	More than 8 unacknowledged blocks are now outstanding for one terminal.	This situation should not arise. See your CDC representative. This problem can be circumvented by using LS=0 on all term cards.	STIM
TOO MANY SCRIPT NAMES	A HASPCR or HASPLP input card contains too many script name parameters, or they specify more devices than are contained in the LCF.	Correct problem and rerun.	STIM
TOO MANY SCRIPTS ON SCRIPT LIBRARY	SCRIPT library directory record is too large for STIM's buffer.	Reduce size of SCRIPT library and rerun.	STIM
UNKNOWN PARAMETER SYMBOL	Unknown parameter on the specified input card.	Correct error and retry.	STIM
UNKNOWN SCRIPT TERMINAL TYPE	Fatal error.	Correct error and rerun.	SCRIPT
UNRECOGNIZED KEYWORD	Fatal error.	Correct error and rerun.	SCRIPT
/ SEPARATOR MISSING	The / separator is missing on a TERM input card.	Correct error and retry.	STIM
= CHARACTER MISSING	An = character is missing on the specified input card.	Correct error and retry.	STIM
xxxxxx CARD ERROR	Input card xxxxxx has an error specified by a subsequent DFM.	Correct error and retry.	STIM
xxxxxx CARD SEQUENCE ERROR	Input card xxxxxx is out of sequence.	Correct error and retry.	STIM
xxx/HOST RECORD READ ERROR nn	Status error nn returned when reading the xxx NHP files host record.	Correct indicated error nn and retry.	STIM
xxxxxxx LOGON ERROR	The abbreviated logon procedure was unsuccessful for terminal xxxxxxx. User name cannot be inserted into the upline message.	Correct problem and rerun.	STIM
xxxxxxx - NO TIMEOUT JUMP ADDRESS	Terminal named xxxxxxx has timed out waiting for response from the host without a specified timeout process declared in the script.	None.	STIM

TABLE B-1. DIAGNOSTIC MESSAGES (Contd)

Message	Significance	Action	Issued By
xxx/NPU RECORD READ ERROR nn	Status error nn returned when reading the xxx NHP files NPU record.	Correct indicated error nn and retry.	STIM
xxxx OUT OF RANGE	The xxxx parameter is out of range on the specified input card.	Correct error and retry.	STIM
xxxx PARAMETER MISSING	The xxxx parameter is missing on the specified input card.	Correct error and retry.	STIM
xxx SCRIPT EXITED BY TERMINAL xxxxxxx	The terminal named xxxxxxx performed an EXIT from script named xxx.	None.	STIM
xxxxxxx TERMINAL FAILED	Caused by =TFAIL directive.	Recovery is automatic.	STIM
xxxxxxx TERMINAL RECOVERED	Self-explanatory.	Proceed as normal.	STIM
xxxxxx - TIMED OUT	Terminal named xxxxxx has timed out waiting for response from the host.	None.	STIM
1ST VALUE GE 2ND	On the DELAY card, the first value must be less than the second.	Correct specification and rerun.	STIM

GLOSSARY

This glossary contains terms unique to the description of the network software, and those common terms whose interpretation within this manual is intended to be more constrained or different from that commonly made.

ADDRESS INFORMATION -
The information within a block or message that identifies the source or intended destination of the associated data.

ALPHANUMERIC -
Unless otherwise specified, the terms alphanumeric, alphanumeric character, or character all imply any of the characters in the CDC 64-character Graphic Display Code character set.

APPLICATION -
A program resident in a host computer that provides an information storage, retrieval, and/or processing service to a remote user via the data communications network.

APPLICATION INTERFACE PROGRAM (AIP) -
Run-time subroutines that reside in the application program's field length to translate and buffer calls made by the program to the Network Interface Program.

APPLICATION NAME -
Up to seven 6-bit letters or digits (the first must be a letter) used to identify an application program. The application name is used by another application program or by a terminal operator when connection to the application is requested.

APPLICATION PROGRAM -
see Application.

ASYNC -
Asynchronous terminal protocol.

BATCH -
All input is entered in a single stream, usually through a card reader in the form of punched cards, and runs to completion with no further external interference allowed. Contrast with interactive.

BLK -
A portion of a message block, but not the last portion. See MSG.

BLOCK -
A portion or all of a message. A message is divided into blocks to facilitate buffering, transmission, error detection, and correction of variable length data streams.

The term block has several meanings depending on context. On tape, a block is information between interrecord gaps on tape.

BLOCKING -
The process of dividing a contiguous data stream into units of generally fixed length.

BREAK -
A method employed by a terminal operator to indicate the desire to interrupt output or input in progress. Also, an element of the block protocol that indicates an interruption of the data stream.

BUFFER -
Consecutive bytes of 2550 Communciations Controller storage used to hold a portion of an information stream.

BYTE -
A group of bits. Unless prefixed (for example, a 6-bit byte), the term implies 8-bit groups.

CHARACTER -
Unless otherwise specified, references to characters are to CDC display-coded 6-bit byte characters.

COMMAND -
Information passed to a process that performs control of the process rather than being data destined for transmission by the process.

COMMUNICATIONS SUPERVISOR -
A portion of the network software executing with NAM as a privileged application program. The Communications Supervisor coordinates the network-oriented activities of its host computer and all of that host's communcation elements.

CONTROL INFORMATION -
Information that is not part of a message, but which must be transmitted in support of the communications protocol.

CONTROLLER -
A hardware device that interfaces multiple terminals to a single communication line and performs some common functions for those terminals (such as protocol handling).

DATA -
Any portion of a message created by the source, exclusive of any information used to accomplish transmission of such a message.

DEADSTART -
The process of loading and starting a processor.

DESTINATION -
The terminal or application program that is designated to receive the message.

DEVICE -
An input-only or output-only portion of a terminal.

DIALOG -
The message interaction between a terminal and the network application running in the host computer.

DIRECTIVE -
All SCRIPT declarations used in preparing scripts are called directives in this publication. All script directives have = as the first character.

DISCONNECT -
The state transition that occurs when a terminal connected to the terminal node via the switched network ceases to be so connected, because its modem goes "on hook," or because of a failure of the switched network.

DOWNLINE -
The direction of output flow, from host to terminal.

FRONT-END -
A network processing unit that directly interfaces one or more hosts.

GLOBAL DIRECTIVE -
A directive that remains in effect in one script until satisfied or cancelled. Contrast with local directive.

HASP STATION -
A terminal that supports the HASP (Houston Automatic Spooling Program) protocol. Examples of such terminals are given in the Network Definition Language Reference Manual.

HOST -
A digital computer that executes application programs.

INFORMATION -
A one-dimensional stream of bits that is communicated from one point to another, exclusive of synchronizing patterns that establish the sample point for the receiver.

INOPERATIVE -
Not operational.

INPUT -
Information flowing upline from terminal to host computer.

INTERACTIVE -
A terminal is said to be interactive when the terminal user can interact with an executing program. Contrast with batch.

INTERACTIVE VIRTUAL TERMINAL (IVT) -
The interactive virtual terminal concept is intended to overcome the differences between the output of various terminals. In particular, IVT helps overcome differences in output formats that could produce undesired or unintended results.

LOCAL CONFIGURATION FILE (LCF) -
A file in the local host computer system, containing information on the physical makeup of the local system. The file contains a list of the application programs available for execution in the local host computer, and the lines and terminals that can access it.

LOCAL DIRECTIVE -
A directive that is executed only when it is encountered. Contrast with global directive.

LOCAL OPERATOR (LOP) -
The local operator manages the communication elements of the network within the local computer system by communicating with the Communications Supervisor in the local host computer. Contrast with network operator. The local operator might also be the host computer's console operator.

MESSAGE -
A logical unit of information as processed by an application program. When transmitted over a network, a message can consist of one or more physical blocks.

MODE -
The mode of a script must be specified as either interactive or batch. The mode must be respecified in the script each time the following directives require such change.

MODE 4 -
This communication line transmission protocol requires the polling of sources for input to the data communication network. Control Data supports two types of mode 4 equipment: mode 4A and mode 4C. Mode 4A equipment is polled through a single hardware address (usually that of the console device), regardless of how many devices use the address as the point of interface to the network. Mode 4C equipment is polled through several hardware addresses, depending on the point each device uses to interface with the network.

MSG -
A message block that is the last portion, or all, of a message. See BLK.

NETWORK ACCESS METHOD (NAM) -
A software package that provides a generalized method of using a communication network for switching, buffering, queuing, and transmission of data.

NETWORK CONFIGURATION FILE (NCF) -
A network definition file in the network control center host computer, containing information on the network elements and permissible linkages between them.

NETWORK DEFINITION LANGUAGE (NDL) -
The compiler-level language used to define the network configuration file and local configuration file contents.

NETWORK HOST PRODUCTS (NHP) -
All of the network products together comprise the Network Host Products. These include the Network Access Method (NAM), Network Definition Language (NDL), Remote Batch Facility (RBF), and so forth.

NETWORK OPERATOR (NOP) -
The network operator manages the hardware, linkages, and other network elements of the entire communications network by communicating with the Network Supervisor in the network control center host computer. Contrast with local operator. The network operator can also be a local operator, and the console operator for the host computer at the network control center.

NETWORK PROCESSING UNIT (NPU) -
The collection of hardware and software that supports a set of one or more directly connected computer memory modules that buffer and transmit data between terminals and host computers.

NETWORK PRODUCTS STIMULATOR (NPS) -
A set of three utilities, which together, test and measure network host system software and network applications.

NETWORK SUPERVISOR -
A portion of the network software, executing with NAM as a privileged application program. The Network Supervisor coordinates all of the NPUs in the communication network.

OUTPUT -
Information flowing downline from host to terminal.

PERIPHERAL INTERFACE PACKAGE (PIP) -
The interface package between the PPU of the CYBER computer and the network application.

PERIPHERAL PROCESSOR UNIT (PPU) -
The hardware unit within the CYBER 170 or CYBER 70 computer that performs physical input and output through the computer's data channels.

SIMULATION -
To create the effect of something, such as a compiled script can simulate a terminal.

SOURCE -
The terminal or host computer program that created the message.

STATION -
A provider and/or recipient of data messages; usually synonymous with terminal.

STIMULATION -
To cause terminal activity, thus stimulating network application responses.

STIMULATOR INTERFACE PACKAGE (SIP) -
The interface package between PIP and the STIM utility.

SUPERVISORY MESSAGE -
A message block not directly involved with the transmission of data, which provides information for establishing and maintaining an environment for the communication of data between the application program and NAM, and through the network to a destination or from a source.

TERMINAL -
An entity, external to the communication network but connected to it via a communication line, which supplies input messages to, and/or accepts output messages from, an application program. A terminal can comprise more than one input or output device.

TERMINAL INTERFACE PROGRAM (TIP) -
The collection of software, resident in an NPU, that controls the transfer of messages between a communication line (to which is connected one or more terminals and/or clusters of terminals) and the NPU.

TERMINAL OPERATOR -
That person who is operating the controls of a terminal. Contrast with user.

TERMINAL SESSION -
The entire dialog between a terminal operator and the host computer, from log-in to logout.

THINK-TIME -
A period of time with no activity that a terminal operator would normally require to determine the next desired input.

TIMEOUT -
A state that occurs if there is no message traffic within the specified time limit. Timeout usually causes a terminal to disconnect.

TIP-TYPE -
The Terminal Interface Package support is limited to three terminal types: MODE4, ASYNC, and HASP. These terminal types are also known as TIP-types.

UPLINE -
The direction of input flow from terminal to host.

USER -
That person or group of people who are the receivers or preparers of messages communicated with an application program via the network. Contrast with terminal operator.

NPS STATEMENT SUMMARY

NPS statements are structured as keyword identifiers followed by parameters. The identifiers require at least one parameter. The statement name, format, function, and page number are shown in table D-1.

TABLE D-1. NPS STATEMENT SUMMARY

Statement Name	Format	Function	Page Number
HASPCR	{HASPCR / HC} (sss_1,...,sss_n)	Identifies the scripts that are to be used to simulate HASP card reader input and identifies the number of card readers to be simulated on a particular HASP station.	3-5
HASPLP	{HASPLP / HL} (sid_1=$trmname_1$,...sid_n=$trmname_n$, sidx/N)	Identifies the scripts that are to be used to process output file data received from the host system and identifies the terminals that are to be configured and expected to receive the output.	3-5
Input Speed Delay	IS {(isx) / (nt1/is1,nt2/is2,...ntn/isn)}	Specifies the input typing speed in characters per second for interactive terminals.	3-3
LNODE	{LNODE / L} (n)	Identifies which frontend NPU in the local configuration file is to be configured for simulation by STIM.	3-4
REPORTR Call	REPORTR(LF=lfn,D1=nnnnn,D2=mmmmm,T1=sssss, T2=ttttt,DM=y,R=iJk,...L=ofn)	Specifies the parameters used in the REPORTR program.	4-1
RNODE	{RNODE / R} (n)	Identifies which remote NPU in the local configuration file is to be configured for simulation by STIM.	3-4
SCRIPT Call	SCRIPT(N=nsl,I=ifn,L=ofn,LO=opt)	Specifies the input/output options desired in the SCRIPT Program.	3-1
STIM Call	STIM(SL=sln,LF=lfn,ET=nnnn,I=ifn,LC=lcf,NC=ncf)	Specifies the parameters used for the STIM run.	3-3
TERM	{TERM / T} (LN=$linename_1$,TN=$trmname_1$/ LN=$linename_2$,TN=$trmname_2$, SN=sid,LG=nnnn,UN=user name, LS=line speed)	Identifies terminals configured in the local configuration file tables, which are to be selected by NPS for simulation.	3-4
Think-Time Input	{DELAY / D} (p)	Establishes a think-time applicable to all the scripts for an entire test run.	3-3

INDEX

CONTROL DATA CORPORATION

COMMENT SHEET

MANUAL TITLE: Network Products Stimulator
Version 1 Reference Manual

PUBLICATION NO.: 60480500 **REVISION:** D

NAME: _____

COMPANY: _____

STREET ADDRESS: _____

CITY: _____ **STATE:** _____ **ZIP CODE:** _____

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

CUT ALONG LINE

AA3417 REV. 4/77 PRINTED IN U.S.A.

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND STAPLE

TAPE

TAPE

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Publications and Graphics Division
215 Moffett Park Drive
Sunnyvale, California 94086

CUT ALONG LINE

FOLD

FOLD

TAPE

TAPE