TELAMON

Synchronous Network Engine
Reference Manual

HP 3000 MPE/iX and MPE/V Version

Telamon, Inc.
492 Ninth Street., Suite 310
Oakland, CA 94607-4098
USA
510-987-7700
Sales 888-835-2578

Warranty

Telamon warrants the Synchronous Network Engine to be free from defects in material and workmanship for one year from the date of delivery. Damage due to accident, abuse, or improper operation is not covered by the warranty. Within the first year, Telamon will pay the cost of replacement components and the cost of shipping them to you. You are responsible only for the cost of shipping the failed Engine or Power Unit back to us.

If you call us during business hours, Monday through Friday, with an apparent hardware problem, we will endeavor to ship replacement equipment that day via next-day delivery service. If you call after 3:00 PM (Pacific Standard Time), the replacement will be shipped the following business day. For those customers outside the United States, we will use air freight. After you receive the replacement, you must ship the failed unit back to Telamon, prepaid, within ten (10) business days, or the replacement will be invoiced as a new order.

After the warranty period has expired, Telamon will respond to your notification of a problem in the same manner as described above, but will charge a nominal fee, currently $100 US, per failed Engine or Power Unit ($200 US for customers outside the United States.) As stated above, you are responsible for shipping the failed unit, prepaid, back to Telamon.

No other warranty other than the above is expressed or implied. Telamon is not liable for consequential damages.

Note: Customers with time-critical applications may wish to purchase a spare Synchronous Network Engine. If a component fails, it can be easily swapped with the spare and the failed unit returned to Telamon for repair as described above.

Technical Support

Technical support for Telamon products is included free of charge during the first year of ownership. After this time, an annual software support agreement can be purchased. Customers who decline software support may be charged an hourly rate.

When calling for assistance with this product, please inform us that you have a ***Synchronous*** *Network Engine*, as well as the type of HP 3000 to which you've attached the Engine. Telamon manufactures a number of other Engine products, including an *Asynchronous* Network Engine, and we can better assist you when the computer and Engine types have been identified.

Telamon is located in Oakland, California and is on Pacific Standard Time. Telamon technical support can be reached via:

- 510-987-7700 (Voice)
- 510-987-7009 (Fax)
- support@telamon.com (Internet e-mail)

If you would like information on Telamon's other products, please call us or send email to:

- sales@telamon.com

Notice

The information contained in this document is subject to change without notice.

Telamon shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Printing History

This page contains the printing history of all the pages in this manual. The listing below shows the pages and the dates on which they were printed.

New editions are complete revisions of the manual. Update packages, which may be issued between editions, contain additional and/or replacement pages to be merged into the manual by the customer. The dates on the title page change only when a new edition or a new update is published. The edition does not change when an update is incorporated.

Edition 1.....October, 1994
Edition 2.....April, 1995
Edition 3.....June, 1997

Table of Contents

APPENDIX H *NETTRANS*

CHAPTER 1 Installation

Introduction

The Network/S program is used to allow the HP 3000 to communicate with the Synchronous Network Engine. Together Network/S and the Network Engine provide 2780/3780 file transfer capabilities for the HP 3000. Network/S accepts most of the commands supported by Hewlett-Packard's RJE product. In addition, the program contains many useful command extensions to the Remote Job Entry command set.

Installing the Network Engine and Network/S software is normally an easy process that should take no longer than 30 minutes. This chapter addresses the installation process. The Network/S software also contains online help. Instructions to access the online help file are included below.

Installation Requirements

- MANAGER.SYS log on
- Familiarity with setting up HP 3000 serial ports
- 20,000 sectors of disc space
- 2 EIA-232 (RS-232) 25-pin cables, preferably shielded, no longer than 50 feet. Specific pin wiring requirements are discussed below.
- Usually a modem is also required. This modem must be able to perform synchronous communication (as opposed to asynchronous communication). It is often desirable for the modem to have autodial capability where phone numbers are passed into the modem from the local computer (as opposed to stored within the modem).

Software Installation

Network/S is stored on a magnetic tape containing the program and associated files. To install the software, mount the tape on your tape drive and type the following commands:

```
:HELLO MANAGER.SYS
:FILE TELAMON;DEV={TAPE DEVICE}
:RESTORE *TELAMON;TELAMON.PUB.SYS;SHOW
```

When the **TELAMON** file has been restored, type:

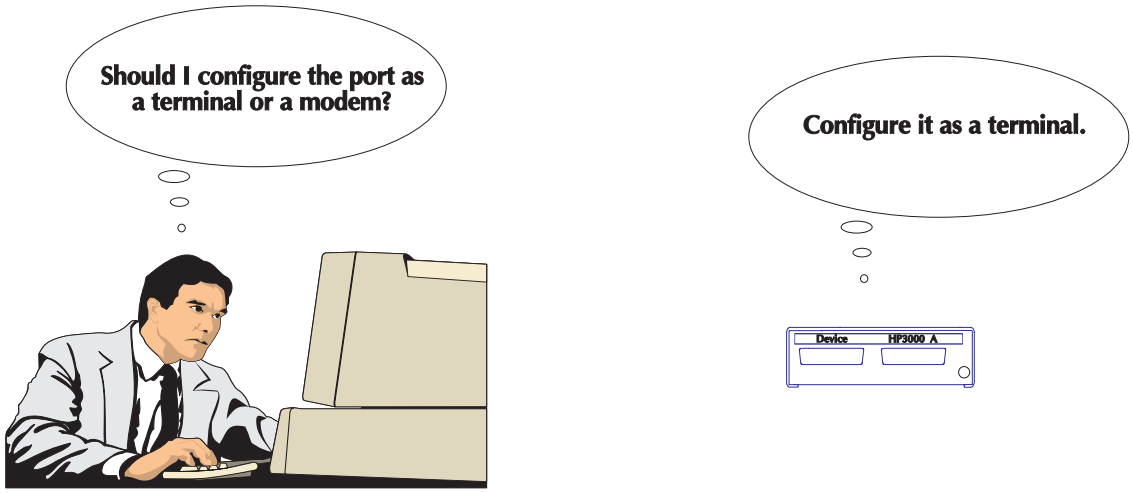
```
:RUN TELAMON.PUB
```

This program will prompt you for various user and account passwords and the *tape device* specified in the **:FILE** equation above. The **MANAGER.SYS** passwords are your existing passwords and the **MANAGER.TELAMON** passwords will become the new passwords for **MANAGER.TELAMON**. The program will prompt you to assign them. The program will create the job streams necessary to build and/or re-configure the **TELAMON** account and to restore the files from your installation tape.

When the stream file has been created, the program will prompt you again to confirm whether you want to stream the job. Enter Y to stream the job and N to suppress the stream. If you enter Y, the job will be streamed and the stream file will be purged. If you enter N, you will be prompted to save the created job stream. If you enter Y, the job stream will be saved as **$OLDPASS** allowing you to view or manually stream the job. If you enter N, the job stream will be purged.

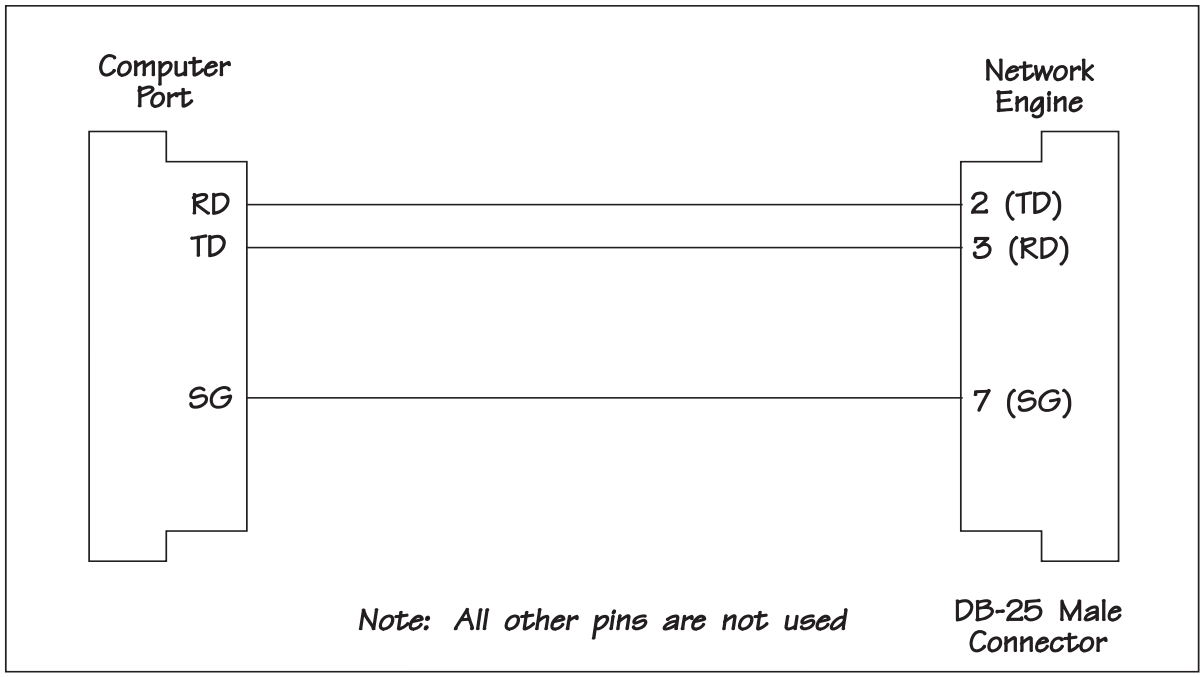
HP 3000 Port Configuration

The Synchronous Network Engine is designed to be connected to an EIA-232-C (RS-232-C) terminal port on either a Classic HP 3000 (MPE/V: ATC, ADCC, or ATP terminal controller) or a Precision Architecture HP 3000 (MPE-iX: DTC terminal controller). The only requirement is that the port be configured as direct connect. On Classic HP 3000's, this means Device-Type 16 and Sub-Type 0 or 4. On the Precision Architecture HP 3000's, this means that the Modem Type field in the DTS Terminal Profile screen must have a value of 0 (e.g. no modem). If you are using a PC-based DTC, the ATTACHED DEVICE = TERMINAL.



Connecting the Network Engine to the HP 3000

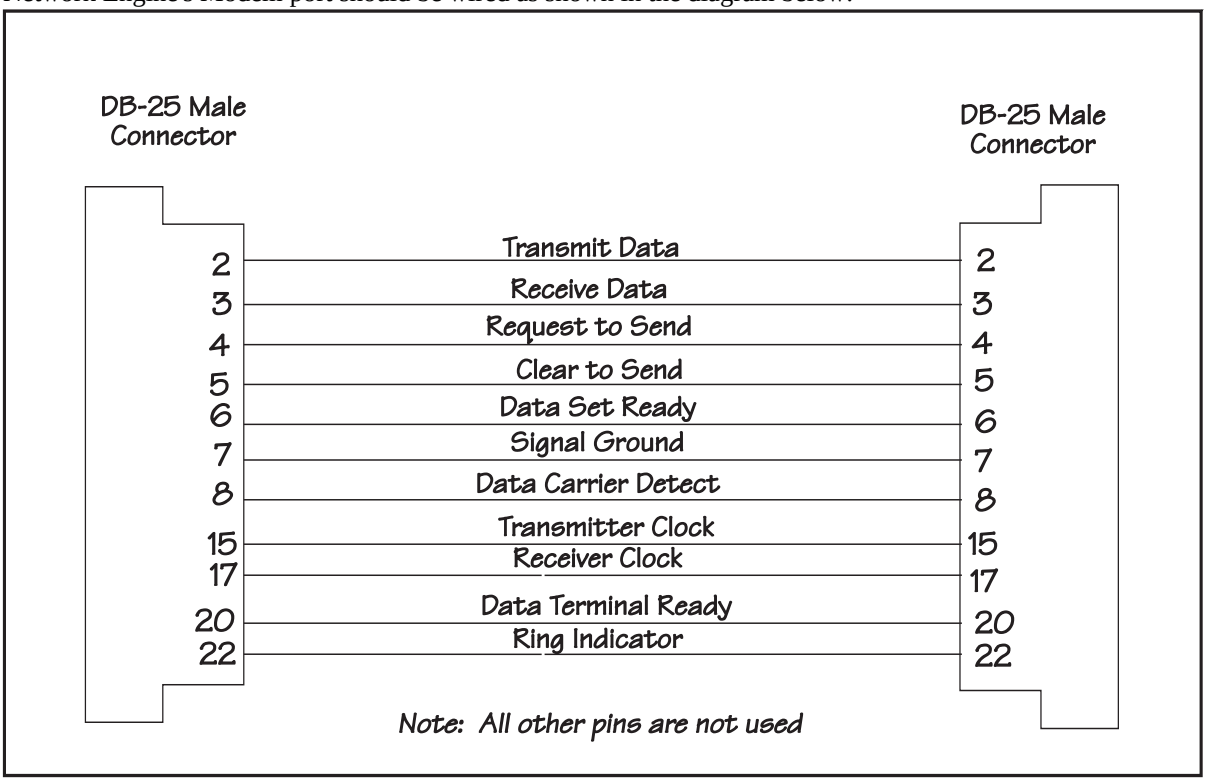
The cable between the HP 3000 and the Network Engine's HP 3000 port should be wired as shown in the diagram below:



The Network Engine is an RS-232 DTE device - any cable that may be used to connect a terminal to your HP 3000 may be used to connect the Network Engine as well. A cable containing all 25 pins wired straight-through (no crossovers) may be used.

Connecting the Network Engine to the Modem

If the Network Engine will be connected to a modem or modem eliminator, the cable between the modem and the Network Engine's Modem port should be wired as shown in the diagram below:



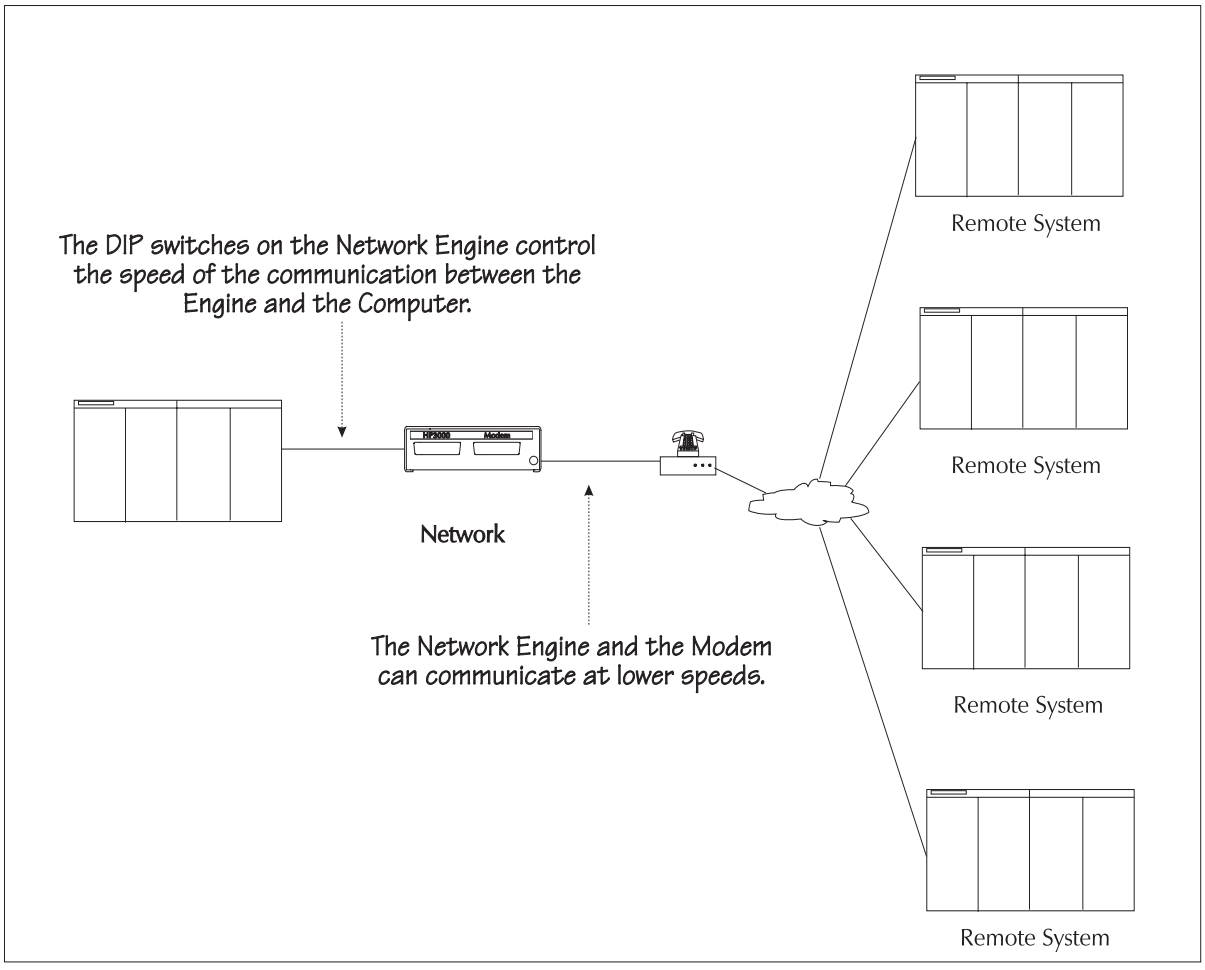
Modems and modem eliminators provide transmit clocking signals on EIA-232 (RS-232) pin 15 and receive clocking signals on pin 17.

If you are connecting the Network Engine to a DTE device instead of a modem, see the additional notes at the end of this chapter.

If you are using an 801 auto-call unit, see the additional notes at the end of this chapter.

Setting the Network Engine's DIP Switches

The 8-position DIP switch on the Network Engine is used to configure the asynchronous baud rate between the HP 3000 and the Engine. This baud rate refers *only* to the rate at which the HP 3000 and the Network Engine communicate. It is completely unrelated to the speed at which the Network Engine communicates with the modem except that it should be *higher* than the modem's speed. For example, the DIP switches may be set at 9600 baud allowing the HP 3000 and the Network Engine to communicate at 9600 baud while the modem may be configured to transmit as 2400 or 4800 baud. The Network Engine contains its own memory buffer freeing the HP 3000 from being bound by the modem's transfer rate. When possible, set the DIP switches to communicate at 19200 baud.



The DIP switch settings are as follows:

TABLE 1-1.

1-4	**Computer Baud Rate**
UDUD	2400
UUDD	4800
UUUD	9600
UUUU	19200
5-8	**Unused**
DDDD	Default

On Dual Engines, the left switch controls the HP 3000B port and the right switch controls the HP 3000A port.

Applying Power To Your Network Engine

When you plug in your Network Engine, the red power light near the Network Engine's DIP switches should come on. If you are in an area that experiences thunderstorms, please consider connecting the Network Engine to a surge suppressor. Each year at Telamon, we receive a number of returned Engines with burned components and blackened printed circuit boards. Protect your Engine as you would your other computer equipment critical to your business.

Running Network/S

All of the Telamon-supplied files are found in the **NETWORKS** group of the **TELAMON** account. Unless otherwise indicated, all filename references to follow will be assumed to be in this location. For the remainder of this document, the group and account names will be omitted.

An MPE/iX Native Mode version of Network/S is available in the NETWORKS group. This version, NETNM, is completely compatible with the Compatibility Mode version, NETWORK. If you'd like to use this version, rename it as follows:

```
:RENAME NETWORK.NETWORKS,NETCM.NETWORKS
:RENAME NETNM.NETWORKS,NETWORK.NETWORKS
```

Note: If you have used SL procedures (**#RJIN**/**#RJOUT**/**#RJCONTINUE**) with either a previous version of Network/S or with Hewlett Packard's **RJE** product, and you intend to use the Native Mode version of Network/S, you *must* convert these procedures to Native Mode and place them in an XL. The Native Mode version of Network/S does *not* utilize procedures found in Compatibility Mode SLs.

The Network/S program can either be run from the **TELAMON** account or it can be placed in **PUB.SYS** and named **RJE**. To run the program from the **TELAMON** account, enter:

```
:RUN NETWORK;INFO="[cmd][,[inp][,[lst][,pnch]]]"
```

Network/S can be installed as **RJE.PUB.SYS** to make use of MPE's built-in recognition of the **:RJE** command. To install the program this way, enter:

```
:HELLO MANAGER.SYS
:FCOPY FROM=NETWORK.NETWORKS.TELAMON;TO=RJE.PUB;NEW
```

or:

```
:COPY NETWORK.NETWORKS.TELAMON,RJE.PUB
```

To run this version, enter:

```
:RJE [cmd][,[inp][,[lst][,[pnch]]]
```

Note: Telamon does *not* recommend that you copy Network/S to **PUB.SYS**. While this option can, and does, work successfully, we generally don't suggest installing software in the **SYS** account. If your existing jobs already make use of the **:RJE** command, we recommend instead that you create a system-wide UDC using the **RJE** definition found in the **MPEVUDC** file.

If you are installing Network/S on an MPE/iX based system, you can use the **MPEXLCMD** command file to run the Network/S program using HP's RJE parameter passing conventions. To setup this command file, enter:

```
:HELLO MANAGER.SYS
:COPY MPEXLCMD.NETWORKS.TELAMON,NERJE.PUB
```

If you use the **HPPATH** system variable to define a common group and account for your command files and programs, copy the **MPEXLCMD** file to the specified location. To run Network/S using this technique, enter:

```
:NERJE [cmd][,[inp][,[lst][,pnch]]]
```

The only way to make direct use of the system **:RJE** command under MPE/iX is to copy the Network/S program file to **RJE.PUB.SYS**. Because **RJE** is a built-in sub-system command, it cannot be re-specified using a command file although it can be redefined by a User Defined Command (UDC).

For MPE/V systems, we have included a UDC file, **MPEVUDC**. Enable the UDC file for the users and/or accounts as is appropriate for your site.

MPE Feature Note

MPE/iX and MPE/V share a feature regarding the reading of commands from **$STDINX** in Batch jobs. When an application has been started by way of an explicit or implicit **!RUN** command or via a built-in MPE command and that application explicitly opens **$STDINX** to read commands and/or data, upon program termination the operating system will *automatically* flush non-MPE commands (those lines not starting with the "**!**" character) from **$STDINX** until the next apparent MPE command is encountered. This means that, in the event of an error during the execution of Network/S, unread **#RJ…** commands will be *flushed* from the job stream and, if a **!CONTINUE** statement has preceded the **!RUN** command, MPE will continue the job with the MPE commands that follow.

If, on the other hand, Network/S has been invoked using either a UDC definition or an MPE/iX command file, this flushing feature is *not* provided.

If you want to take advantage of this flushing feature, you will need to invoke Network/S either by way of an explicit **!RUN** command, by copying the software to **RJE.PUB.SYS** or, under MPE/iX, by copying the software to some location on your path (specified using the **HPPATH** system variable) and invoking the application via the implied run mechanism.

The Network/S Help Sub-system

Network/S utilizes the Qhelp sub-system kindly provided by Robelle Consulting, Ltd. The Help subsystem is invoked from within the Network/S program and the Help file cannot be read except by applications that can view Qedit-format files.

To access the Help subsystem, run the Network/S program and at the prompt, enter:

```
#HELP
```

To print the entire help file to your line printer, enter the command

```
#PHELP @
```

instead. The formal file designator, **QHELPOUT**, may be used to redirect or prioritize the printing.

Configuring Your Modem

The most difficult part of the installation is often configuring your modem to communicate in bisync and to auto-dial. Please refer to the help file (**#HELP CONFIG,MODEMS**) for information on modem configuration. The modems listed have been tested at Telamon. If your modem isn't included in the help section, you will find general configuration information at the end of the help section. Detailed modem configuration information can also be found in the *Modem Configuration* Appendix.

*Building an **RJECLINE** File*

Network/S provides a means to override the normal default values that apply when the program is run. At run-time, Network/S will search for a configuration file. If found, the file's contents will be read and used to override various program defaults. Network/S will search in a number of locations for this file. They are:

```
RJECLINE.logongroup.logonaccount
RJECLINE.PUB.logonaccount
RJECLINE.PUB.SYS
RJECLINE.NETWORKS.TELAMON
```

If one of the above files is found, it will be read from start to finish. Only the first file found will be read. See “Defaults” on page 4-1.

Valid **RJECLINE** commands include (defaults, when indicated, are underlined):

```
LINECODE={ASCII | EBCDIC}  [, {CRC | LRC}]
MODEM={DTR | UDS | SADL | V.25[,ASCII] | 801}
SELECT={MODEMA | MODEMB}
DEV=[logical device # of Engine port] [,speed of DEV in cps]
PRI={HIGH | NORMAL}
BAUDRATE=[synchronous speed, if provided by Engine]
SHOW={YES | NO}
VERBOSE={YES | NO}
WARN={YES | NO}
CONSOLE={YES | NO}
TRACE={YES | NO}
CRASH=n
SYN=n
ENQ=n
NAK=n
TTD=n
```

Thus, to change the default **LINECODE** to **EBCDIC** (from **ASCII**), you would enter:

```
LINECODE=EBCDIC
```

in the **RJECLINE** file.

If you have attached your Network Engine to logical device 32 and you don't want to reconfigure that port to have a device class name of **RJLINE**, you could enter:

```
DEV=32
```

in the file.

If no **RJECLINE** file is found, Network/S proceeds with the normal defaults.

Multiple Synchronous Network Engines

If you have more than one Synchronous Network Engine, you can create defaults files for specific devices. The initialization process is:

- When Network/S is run, it searches for an **RJECLINE** file in the locations and order listed above.
- After an **#RJLINE** command is issued, Network/S will search for a file named **RJECL*nnn*** where *nnn* is the LDEV number of the device specified with the **DEV=** parameter of the **#RJLINE** command. Network/S will then read this file for default parameters for this port.

Example:

The Acme Explosives company has two Synchronous Network Engines. On one Engine, a Racal-Milgo RMD3222 modem is attached. On the other Engine, a UDS 208B/D modem is attached.

An **RJECLINE** file can be created storing Acme's site defaults:

```
LINECODE=EBCDIC
VERBOSE=YES
```

When ACME runs Network/S, the **LINECODE** and **VERBOSE** parameters are read from the **RJECLINE** file. When ACME issues the **#RJLINE** command, Network/S will search for a file called **RJECL030** in the following example:

```
#RJLINE 3780;DEV=30,1920;CONNECT=DIAL,"915105551212"
```

If the Racal-Milgo RMD3222 modem is attached to the Engine on LDEV 30, ACME will have at least the following parameter in their **RJECL030** file:

```
MODEM=V.25,ASCII
```

The Racal-Milgo RMD3222 modem is dialed using the V.25 autodial commands sent to it in ASCII. If the UDS 208B/D modem is attached to LDEV 107, ACME will have the following parameter in the **RJECL107** file:

```
MODEM=UDS
```

Script Branching

With the use of script branching, you can have your command file or job use an alternate phone number if one line is busy. For example:

```
#RJCONTINUE NOABORT
#RJLABEL DIALRMD3222
#RJLINE 3780;DEV=30,1920;CONNECT=DIAL,"915105551212";NOW
#RJIF !CSERROR = 0 THEN GOTO SENDLOGON
#RJCOMMENT +++ CSERROR 11 INDICATES THAT THE DEVICE IS
#RJCOMMENT +++ UNAVAILABLE
#RJIF !CSERROR = 11 THEN GOTO DIALUDS
#RJCOMMENT +++ IF SOME OTHER PROBLEM IS ENCOUNTERED THEN
#RJCOMMENT +++ QUIT
#RJGOTO QUIT
#RJLABEL DIALUDS
```

```
#RJCOMMENT +++ LDEV 107 HAS THE UDS MODEM ATTACHED TO IT.
#RJCOMMENT +++ A DIFFERENT PHONE NUMBER IS USED SINCE THE
#RJCOMMENT +++ UDS MODEM USES THE BELL 208 PROTOCOL
#RJCOMMENT +++ (4800bps) INSTEAD OF THE V.32 PROTOCOL
#RJCOMMENT +++ CONFIGURED IN THE RMD3222.
#RJLINE 3780;DEV=107,1920;CONNECT=DIAL,"915105551234";NOW
#RJIF !CSERROR <> 0 THEN GOTO QUIT
#RJLABEL SENDLOGON
#RJIN MYLOGON...
.
.
.
#RJEND
#RJLABEL QUIT
#RJABORT
```

Installation Checkout

Using a text editor, create or modify the file **RJECLINE** to specify the port number to which the Engine is connected, the speed at which the port will communicate with the Network Engine (see the DIP switch table earlier in this chapter), and the type of autodialer your modem uses.

Example:

```
DEV=101,1920
LINECODE=EBCDIC
MODEM=V.25,ASCII
```

Keep the file *unnumbered* as RJECLINE.NETWORKS.TELAMON.

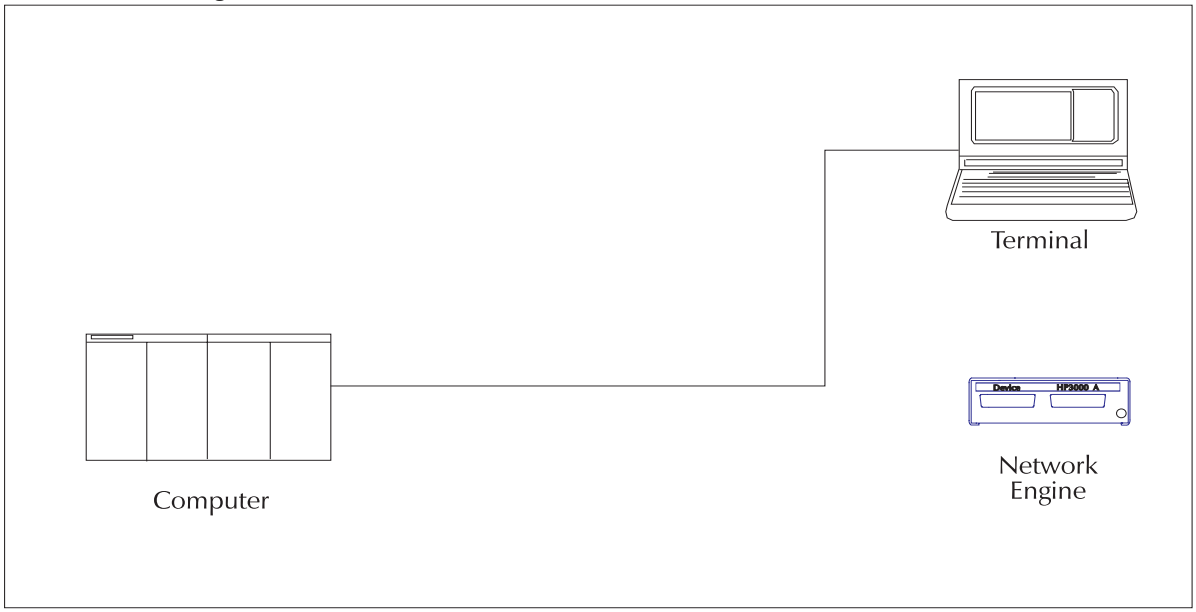
You can test the Computer to Network Engine port connection now by running Network/S, which should give you the following response (you type #**RJLINE** and #**RJEND** from within Network/S):

```
Network/S - Telamon, Inc....
Version ...
Licensee: ...
#RJLINE 3780
RJLINE Settings
  3780;…
Network/S: Engine Version<version> (<version info>)
#RJEND
```

If you don't see the Network/S Engine Version but instead receive a “No Response from Network Engine” message, then proceed to the next page to determine the problem.

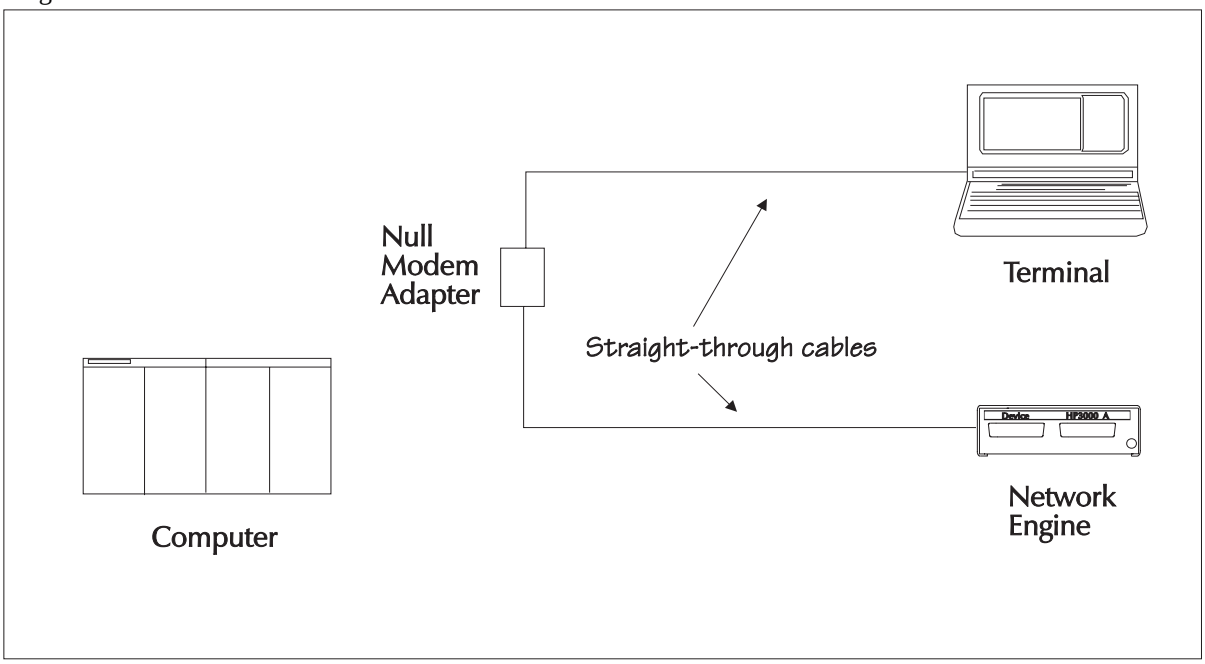
Troubleshooting: What to Do if You Do NOT Get a Response

- If you are using a Precision Architecture HP 3000, verify that the Network Engine's DTC port is configured as a *nailed* terminal port. The terminal profile should have the field *Modem Type* set to 0. If you are using a PC-based DTC, the *ATTACHED DEVICE* field set to *TERMINAL*. At this point, you may want to change your configuration to allow log-ons for further debugging of the port.
- Verify that the Network Engine is plugged in and that the power light is on.
- Verify that the Network Engine's DIP switches match the speed specified in the **#RJLINE** command (DEV=port number, SPEED=speed in cps) or in the **RJECLINE** file you may have set up. If the DIP switches have been changed, the Engine must either be reset (by moving the paddle reset lever in the direction of the DIP switches) or powered off and powered back on. Only by one of these actions will the Network Engine re-read the DIP switches.
- Verify that the cable from the HP 3000's Computer port (serial port) is connected to the Engine port labeled "COMPUTER".
- Verify the cables are straight-through cables (*not* a crossover cable that is typically used to connect an HP 3000 port to a modem). The cable connecting the HP 3000 port to the Network Engine *must* have pins 2, 3, and 7 wired straight-through.
- Leaving the cable attached to the serial port, disconnect the Network Engine and attach a terminal as shown in the diagram below:



- Configure the terminal so that the data-comm port communicates with 8 data bits, no parity and receive pacing NONE. Set the baud rate to match the speed of your HP 3000 port. Hit return a few times. If the port has been set up correctly and the cable has pins 2, 3, and 7 wired straight-through, you should receive a log-on prompt. Log-on to the HP 3000 and type :**SHOWME**. Verify the port (LDEV) number you are connected to. Type :**BYE** to log off. You *must* log off to release the port.
- Leaving the cable attached to the serial port, disconnect the terminal and reconnect the Network Engine and try the *Installation Checkout* test again.

- If you are still having a problem getting a response from the Network Engine, leave the cable plugged into the Engine and disconnect the cable from the computer's serial port. Connect the null modem adapter (included with the Engine) to the cable and plug the cable into a terminal as shown in the diagram below:

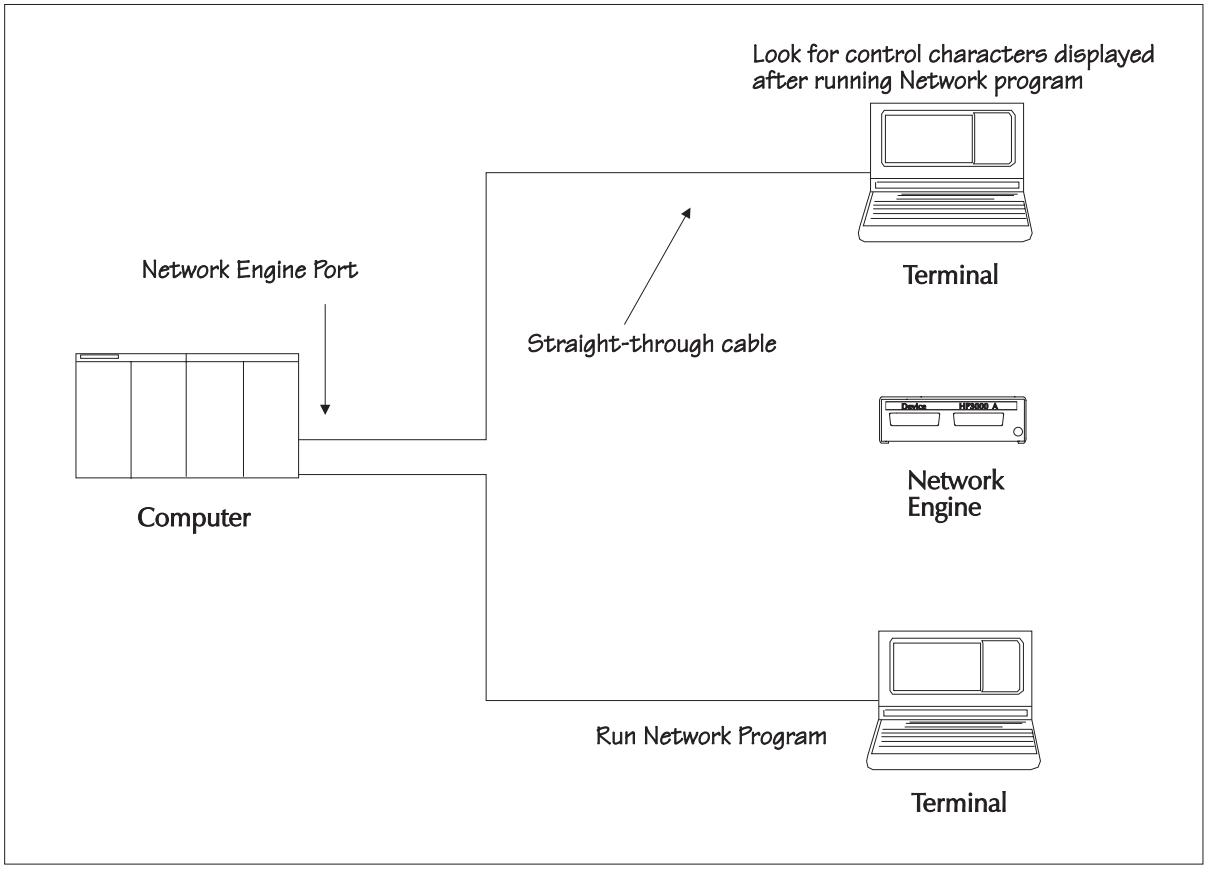


- Configure the terminal so that the datacomm port communicates with 8 data bits and no parity. Turn Display Functions or Monitor mode on. Make sure that the terminal is in Remote Mode. Hold the control key down and type **P** followed by **X** followed by **Q** (control-P, control-X, control-Q.) You should get back a line that looks something like:

```
@@@@@@@@@DZZZ
```

- If you are able to get this response from the Engine, take out the null modem connector, plug the cable back into the computer's serial port, and try running the *Installation Checkout* test again.

- If you are still unable to get a response from the Engine, disconnect the Engine from the computer's cable. Leaving the computer end of the cable plugged in, connect the other end of the cable to a terminal as shown in the diagram below:



- Turn display functions/monitor mode on. Run Network/S from a different terminal and enter your #**RJLINE** command. The terminal should display a series of control characters such as:

```
D C D   D N D
 L N 1   L K 1
```

  If no characters can be seen, the problem must either be with the cable or with the serial port. Re-check the steps above.

- Finally, if none of these tests indicate the source of the problem, contact Telamon Technical Support.

Communications Checkout

You can call Telamon technical support to arrange a Synchronous Network Engine checkout test. We will need to know what speed protocol (i.e. V.32, Bell 208, Bell 201) your modem is configured to use.

To begin the checkout test, run the Network/S program as follows:

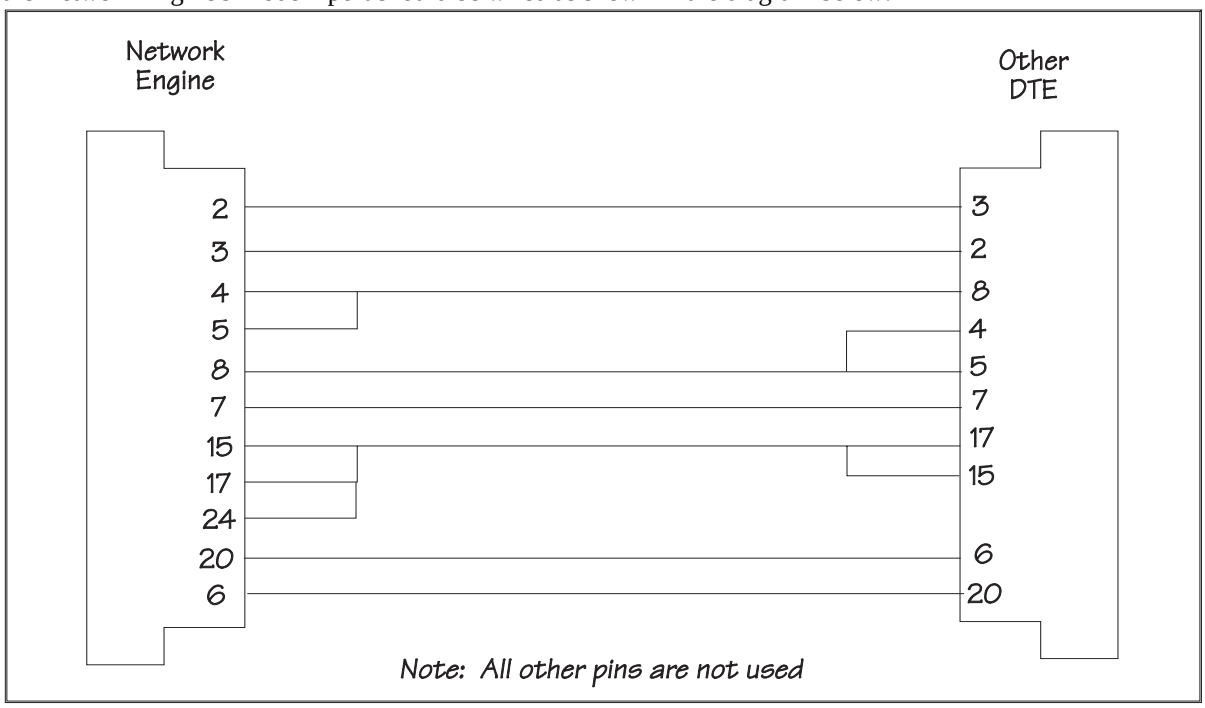
```
:RUN NETWORK;INFO="CHECKOUT.NETWORKS.TELAMON"
```

The checkout test will cause your **RJECLINE** file to be sent to Telamon. If the test is successful, you will receive a message from Telamon on your terminal.

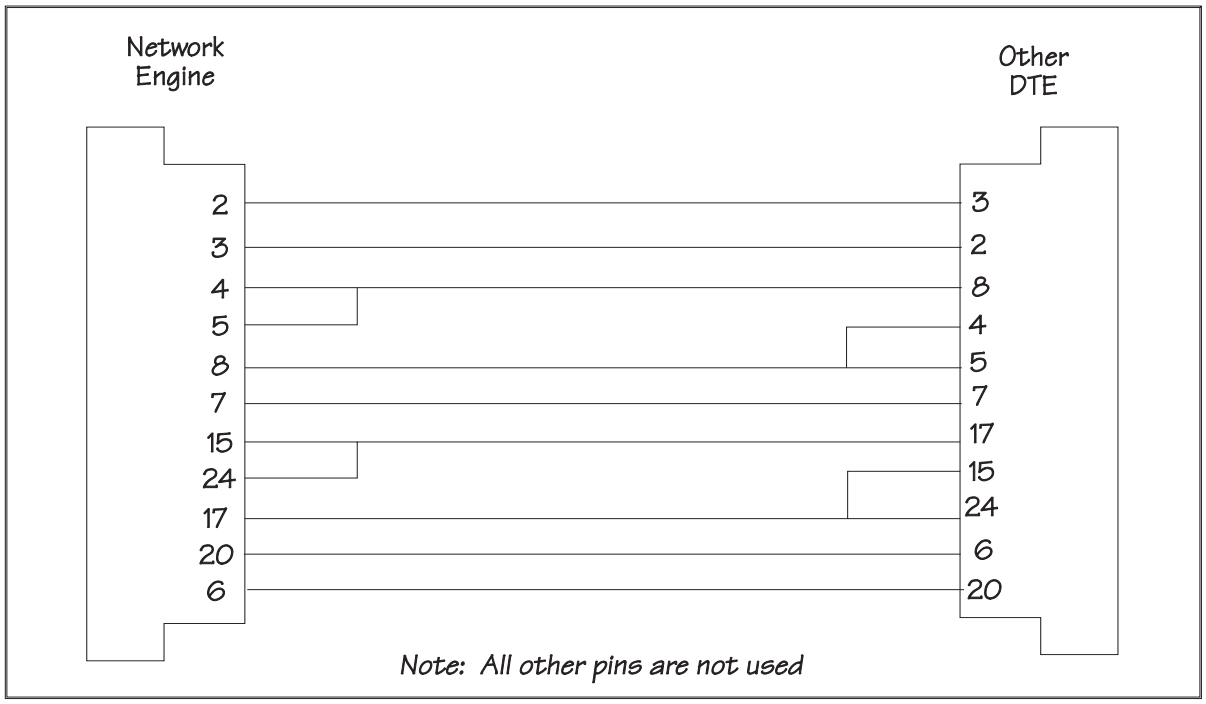
Connecting the Network Engine directly to a DTE Device

The Micro version of the Network Engine can be connected between an HP 3000 and another DTE device without having to use a modem or modem eliminator. In this configuration the Network Engine can provide the necessary clock signals to communicate with the other DTE device.

If the Network Engine is to provide the clock signal for both devices, the cable between the DTE equipment and the Network Engine's Modem port should be wired as shown in the diagram below:

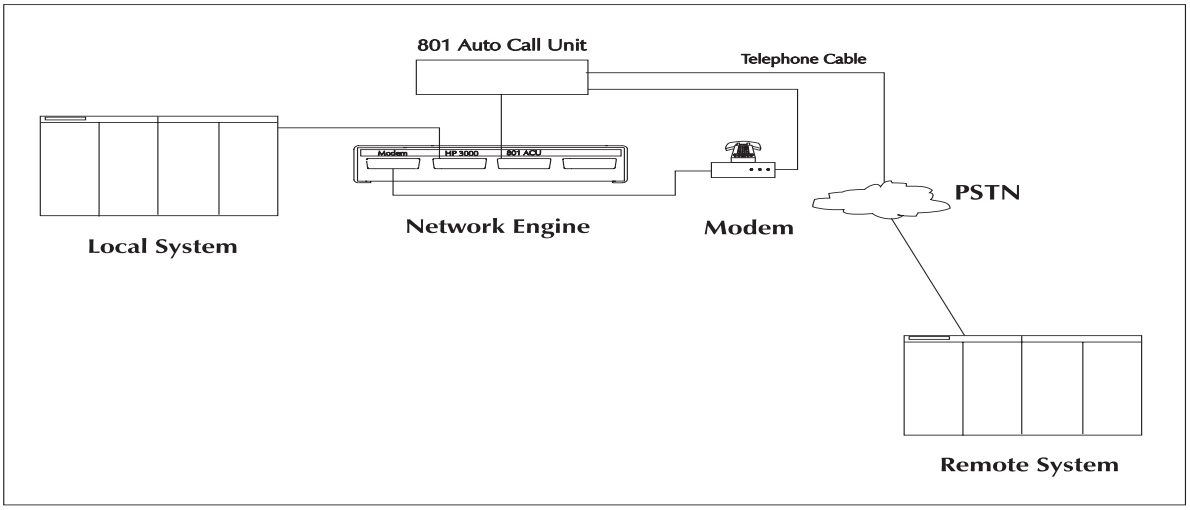


If the other DTE equipment provides a clock signal on EIA RS-232 pin 24 (External Transmitter Clock), the cable between the DTE equipment and the Network Engine's Modem port should be wired as shown in the diagram below:



Using an 801 Auto-Call Unit

If you are using the version of the Network Engine that supports an 801 RS-366 auto-dialer, you will need an additional cable. The cable used to connect the Engine and your computer and the cable used to connect the Engine to the modem are the same as above. The 801 auto-dialer is connected to the Engine on the port marked 801 ACU using a 25-pin straight-through cable.



There are four ports on this version of the Network Engine. The unlabeled port on the Engine is not used. In addition, there are two DIP switches on the Engine. The left DIP switch controls the settings on the Computer port; the right DIP switch is not used.

Please remember to specify the **801** option in the **#RJLINE** command or in the **MODEM** option in the **RJECLINE** file.

CHAPTER 2 Getting Started

RUNNING NETWORK/S

If you have left the Network program in the **NETWORKS** group of the **TELAMON** account, the Network/S software is executed using:

```
:RUN NETWORK.NETWORKS.TELAMON[;INFO="[CMD][,[INP][,[LST][,PNCH]]]]"
```

If you've copied the Network program to the **PUB** group of the **SYS** account and named the program **RJE**, the command line is:

```
:RJE [CMD][,[INP][,[LST][,PNCH]]]
```

This technique passes information about the specified file names using a combination of **:FILE** equations and a **:RUN PARM** value, shown on the table on the following page. Network/S supports this method whether it has been installed in the **SYS** account or left in the **TELAMON** account.

If you are using the MPE/iX command file or the MPE/V UDC file provided with the software, you can use the following command line:

```
:NERJE [CMD][,[INP][,[LST][,PNCH]]]
```

Files

TABLE 2-1.

File	Description	Formal Name	Default	PARM
CMD	File containing commands that Network will execute when it is started.	RJECOM	$STDINX	1
INP	File containing data to be transmitted to remote system (see #**RJIN** command)	RJEIN	$STDINX	4
LST	File on local system that routed listfile output will be written to (see **#RJLIST** and #**RJOUT**)	RJELIST	$STDLIST	2
PNCH	File on local system that routed punchfile output will be written to (see #**RJPUNCH** and #**RJOUT**)	RJEPUNCH	$OLDPASS/ $NEWPASS	8

If either or both of the list or punch files are specified, Network/S first attempts to access each file as an old, permanent or temporary file. If no such file is found, the program creates the file with the following characteristics:

```
:FILE LST;REC=-256,,V,ASCII;CCTL;DISC=5000
:FILE PNCH;REC=40,,V,ASCII;CODE=RJEPN
```

Commands

Commands may be up to 256 characters long and may be spread on multiple lines by use of the standard “&” line continuation character. For example, the following are valid and equivalent commands:

```
#RJLINE 3780;LINECODE=EBCDIC;ID="MYSYSTEM";MAXRPB=3

#RJLINE 3780; &
#  LINECODE=EBCDIC;ID="MYSYSTEM";MAXRPB=3

#RJLINE 3780;LINE&
#  CODE=EBCDIC;ID="MYSYSTEM";MAXRPB=3

#RJLINE 3780;LINECODE=EBCDIC;ID="MY&
#  SYSTEM";MAXRPB=3
```

Standard RJE Commands:

#RJLINE	Define synchronous link's operating parameters
#RJIN	Send input data set to remote host system
#RJOUT	Receive output data set from remote host system
#RJLIST	Receive routed or unrouted print file from remote host system
#RJPUNCH	Receive routed or unrouted data file from remote host system
#RJEOD	Send End-of-Transmission (**EOT**) code
#RJSTAT	Display statistics

#RJIO	Pseudo-interactive communications
#RJINFO	Display configuration information
#RJCMDFILE	Execute command file
#RJCONTINUE	Specify error handling procedure or label
#RJCOMMENT	Insert comment text
#RJEND	Terminate Network/S (Also #**EXIT** and #**END)**

Network/S Extended Commands:

#RJABORT	Force Network/S to terminate in an error state
#RJBID	Perform ID/REMID validation
#RJBOF/#RJEOF	Copy **$STDINX** commands to temp file
#RJDISC	Send shutdown sequence
#RJDISPLAY	Display messages and variables
#HELP	Enter Qhelp subsystem
#PHELP	#**HELP** directed to line printer
#FMTDUMP	Analyze **TRACE** file output
#RJMPE/#RJSYS	Execute MPE command
#REDO	Edit and re-execute Network/S command
#DO	Re-execute previous Network/S command
#RJSHOWTIME	Display current date/time
#RJPAUSE	Pause during Network/S execution
#RJCALC	Perform system calculations
#RJLABEL	Specify branch labels
#RJGOTO	Unconditionally branch to a label
#RJIF	Test variables and conditionally branch on result

Examples

The following example will connect to General Electric's electronic mail service (GEIS) to perform a startup test. This test involves sending a sequence of Job Control Language to GE's host system which requests the transmission of a report summarizing the test. This report is sent by GE as a routed list file and will be spooled to the line printer.

The EBCDIC character set will be used during the synchronous operations and we will be specifying an alternate ASCII-to-EBCDIC character set. Note that the port speed does not indicate the speed of the modem. Most synchronous modems provide their own speed settings - various Universal Data Systems (UDS) modems, for example, can operate at 2400 (201C/D), 4800 (208B/D) and 9600 (3229) bits-per-second.

```
:FILE LP;DEV=LP
:NERJE
Network/S - Telamon, Inc. (C)...
Version...
Licensee...
#RJLINE 2780;LINECODE=EBCDIC;XEND;TABLE=AEIBM.NETWORKS.TELAMON;&
#  CONNECT=DIAL,"geis number",dialer
#RJIN *
#?ANS09102,COMMTEST,MAILA
```

```
#?*LTID MAILBOXA
#?*MODE INPUT,WAIT
#?PRINT MESSAGE;MAILBOXA;NONE
#?*EOS
#?#RJEOD
#RJOUT *LP
#RJEND
```

This sequence can also be implemented by placing the command and data lines into a text file, as follows:

```
#RJLINE 2780;LINECODE=EBCDIC;XEND;TABLE=AEIBM.NETWORKS.TELAMON;&
#  CONNECT=DIAL,"geis number",dialer
#RJIN *
ANS09102,COMMTEST,MAILA
*LTID MAILBOXA
*MODE INPUT,WAIT
PRINT MESSAGE;MAILBOXA;NONE
*EOS
#RJEOD
#RJOUT *LP
#RJEND
```

Put these lines in a file called **GECMD** and Network could then be executed using:

```
:RUN NETWORK;INFO="GECMD"
```

or:

```
:NERJE GECMD
```

The '*****' designation causes the #**RJIN** command to read data records from the current command file, **GECMD** in this case.

The commands are displayed on **$STDLIST** during execution. Note the presence of the # prefix characters on each Network/S command line. When Network reads commands and data from the same file, it must have some means to distinguish between commands and data. In this context, any line beginning with a # prefix is considered to be a Network/S command. In other contexts, the # prefix is optional.

The next example connects to the IBM Information Exchange (Advantis) Network (expEDIte/Direct) in order to perform an EDI data transfer.

```
:NERJE
Network/S - Telamon, Inc. (C)...
Version...
Licensee...
#RJLINE 3780;LINECODE=EBCDIC; &
#  CONNECT=DIAL,"advantis number",dialer
#RJBID
#RJOUT $STDLIST
#RJIN *;COMPRESS=NO;TRUNCATE=NO
#?/*LOGON account,userid,password/*SELECT ...
#?#RJEOD
```

```
#RJOUT $STDLIST
#RJIN *;COMPRESS=NO;TRUNCATE=NO
#?IELOGON ACCOUNT(account)...;
#?#RJEOD
#RJOUT $STDLIST
#RJIN *;COMPRESS=NO;TRUNCATE=NO
#?SENDEDI;
#?#RJIN SENDDATA;COMPRESS=NO;TRUNCATE=NO
#RJOUT $STDLIST
#RJIN *;COMPRESS=NO;TRUNCATE=NO
#?/*LOGOFF
#?#RJEOD
#RJOUT $STDLIST
#RJEND
```

As above, this sequence can also be implemented by placing the command and data lines into a text file. This example uses a similar scheme to identify the source of the data records. If you place the commands in a file, the file's contents would be:

```
#RJLINE 3780;LINECODE=EBCDIC; &
#  CONNECT=DIAL,"advantis number",dialer
#RJBID
#RJOUT $STDLIST
#RJIN *;COMPRESS=NO;TRUNCATE=NO
/*LOGON account,userid,password/*SELECT ...
#RJEOD
#RJOUT $STDLIST
#RJIN *;COMPRESS=NO;TRUNCATE=NO
IELOGON ACCOUNT(account)...;
#RJEOD
#RJOUT $STDLIST
#RJIN *;COMPRESS=NO;TRUNCATE=NO
SENDEDI;
#RJIN SENDDATA;COMPRESS=NO;TRUNCATE=NO
#RJOUT $STDLIST
#RJIN *;COMPRESS=NO;TRUNCATE=NO
/*LOGOFF
#RJEOD
#RJOUT $STDLIST
#RJEND
```

Put these lines in a file called **IBMCMD** and Network/S would then be executed using:

```
:NERJE IBMCMD
```

Automated Redialing

Network/S allows you to automatically redial failed calls by using a parameter specified in the **#RJLINE** command:

```
#RJLINE 3780;CONNECT=DIAL,"95551212",V.25,ASCII;REDIAL=2,1;...
```

In the example above, Network/S will redial two times, one minute apart, if the original dial attempt fails. See the **#RJLINE REDIAL** option, on page 3-65, for more details.

Script Branching

You may branch to labels within scripts.

```
#RJCOMMENT The #RJCONTINUE following these comments causes sub-
#RJCOMMENT sequent commands (#RJLINE) to continue to the following
#RJCOMMENT command even if there is an error. Errors are trapped
#RJCOMMENT following the #RJLINE command. The NOW parameter
#RJCOMMENT was added to the #RJLINE command so that dialing
#RJCOMMENT will occur on the #RJLINE command, not after the
#RJCOMMENT first #RJIN.
#RJCONTINUE NOABORT
#RJLINE 3780;CONNECT=DIAL,95551212;REDIAL=2,1;NOW
#RJCOMMENT IF THE VALUE-ADDED NETWORK CALL DOES NOT GO
#RJCOMMENT THROUGH, HAVE THE TELAMON CONSOLE ENGINE CALL
#RJCOMMENT SANDY AT HOME
#RJIF !CSERROR = 54 THEN GOTO WAKEUPSANDY
#RJIF !CSERROR <> 0 THEN GOTO NODATA
#RJCONTINUE WAKEUPSANDY
#RJIN SIGNON;COMPRESS=NO;TRUNCATE=NO
#RJEOD
#RJIN SENDDATA
#RJEOD
#RJOUT RECVDATA
#RJCOMMENT IF THERE IS NO DATA TO SEND, THESE GUYS JUST
#RJCOMMENT RUDELY HANG UP! IN THAT CASE, WE FCOPY A FAKE RECORD
#RJCOMMENT INTO THE RECEIVE FILE SO WE CAN CONTINUE PROCESSING.
#RJIF !CSERROR = 158 THEN GOTO NODATA
#RJIF !CSERROR = 103 THEN GOTO NODATA
#RJEND
#RJLABEL WAKEUPSANDY
#RJSYS RUN CENGINE.CONSOLE.TELAMON; &
#  INFO="SEND PHONE='5551234',NONE,ACK,HELP!!!"
#RJEND
#RJLABEL NODATA
#RJSYS FCOPY FROM=FAKEDATA;TO=RECVDATA
#RJEND
```

See the #**RJIF**, #**RJGOTO**, and #**RJLABEL** commands in the Commands Chapter for more details.

Receiving EDI Data into 80-Byte Files

Sometimes it is desirable (or necessary) to receive data into fixed record length files. The #**RJOUT REBLOCK** parameter allows you to reblock incoming records based on the record width of the #**RJOUT** file. If we wanted to receive EDI X.12 data into the **RECVDATA** and we wanted **RECVDATA** to be an 80 byte fixed ASCII file, we could modify a portion of the script so it becomes:

```
.
.
```

```
#RJSYS BUILD RECVDATA;REC=-80,16,F,ASCII;DISC=5000
#RJOUT RECVDATA;REBLOCK=LENGTH
.
.
```

The #**RJSYS** line specifies a BUILD command for **RECVDATA**. The **REBLOCK= LENGTH** parameter on the #**RJOUT** command on the next line causes the data to be blocked into 80 byte records.

Receiving Multiple Data Sets into separate Files

Some Value-Added Networks send a variable number of files when you request them to send the contents of your mailbox. The **AUTONUM** parameter has been added to #**RJOUT** to allow multiple data sets to be written to multiple files. The script above can be modified to accommodate this:

```
.
.
#RJSYS BUILD RECV01;REC=-80,16,F,ASCII;DISC=5000
#RJOUT RECV01,12;REBLOCK=LENGTH;AUTONUM=YES;WAIT=0,10
.
.
```

The file **RECVDATA** has been changed to **RECV01** in the file equation and in the #**RJOUT** command. The **#RJOUT** command now contains a count parameter of 12 after the filename indicating that it will receive up to 12 data sets from the VAN. Each successive file will be built to match **RECV01**, but will be numbered consecutively, **RECV02**, **RECV03**, etc. A **WAIT** parameter has also been added to the #**RJOUT** command. This tells Network/S to wait for a maximum of 10 seconds between files. If a line bid is *not* received in 10 seconds, the next command will be executed *without* a timeout error; it will be assumed that the sender has no more files to send.

It is also possible to receive multiple data sets into a single file by specifying the **,count** parameter and omitting the **AUTONUM** parameter. See the #**RJOUT AUTONUM** option, on page 3-88, for more details.

Writing Scripts

Let's say that we want to start communicating with the Advantis Value-Added Network. We contacted Advantis and have a valid account. From reading their documentation, we know the following:

- Advantis requires to two logons, one for their front end processor (**/*LOGON**) and another to login to the Information Exchange (**IELOGON**)
- Before sending data to Advantis, we must first send a **SEND** command, to inform them of our intentions
- Sending a command called **RECEIVE** to Advantis initiates a transfer *from* Advantis *to* us
- Finally, to receive a transaction log from Advantis, we need to send them the command **RECEIVE CLASS(EDILOG)**

From the information above, we can develop the following script:

```
#RJLINE 3780;CONNECT=DIAL,"advantis number",dialer;LINECODE=EBCDIC
#RJIN *;COMPRESS=NO;TRUNCATE=NO
/*LOGON account,userid,password/*SELECT EDIRECT/*USERDATA BSCEDI
```

```
#RJEOD
#RJIN *;COMPRESS=NO;TRUNCATE=NO
IELOGON ACCOUNT(account) USERID(userid) PASSWORD(password);
#RJEOD
#RJOUT $STDLIST
#RJIN *;COMPRESS=NO;TRUNCATE=NO
SEND ACCOUNT(account) USERID(userid) PASSWORD(password);
#RJIN EDISEND;COMPRESS=NO;TRUNCATE=NO
#RJEOD
#RJIN *;COMPRESS=NO;TRUNCATE=NO
RECEIVE CLASS(X12);
#RJEOD
#RJOUT EDIRECV
#RJIN *;COMPRESS=NO;TRUNCATE=NO
RECEIVE CLASS(EDILOG);
#RJEOD
#RJOUT $STDLIST
#RJOUT $STDLIST
#RJEND
```

We run the Network program with the command file above, resulting in:

```
:BUILD EDIRECV;REC=-80,16,F,ASCII
:NERJE IBMT1
Network/S - Telamon, Inc. (C)...
Version...
Licensee: ...
#RJLINE 3780;CONNECT=DIAL,"advantis number",dialer;LINECODE=EBCDIC
RJLINE Settings
  3780; LINECODE=EBCDIC; PRI=NORMAL
  DEV=device,speed; WAIT=0,0
  CONNECT=DIAL,"advantis number",dialer
Network/S: Engine Version: ....
#RJIN *;COMPRESS=NO;TRUNCATE=NO
RJIN Settings - File reference: "*"
  INCODE=ASCII; XPARENT=NO; TRUNCATE=NO; COMPRESS=NO
  Effective MAXRPB=255
  WAIT=3,0
File characteristics:
  IBMT1,OLD;REC=-80,3,F,ASCII;NOCCTL;DISC=???;ACC=IN
03OCT94-09:43:40.1: Waiting for connection
03OCT94-09:43:50.0: On Line (DSR high)
03OCT94-09:43:50.2: Issuing transmit bid
03OCT94-09:43:54.2: Transmit bid acknowledged: starting transfer
#RJEOD
03OCT94-09:43:54.8: Warning: RVI (Reverse Interrupt) Received
03OCT94-09:43:54.8: Transmit complete                        <-----
1 record, 80 bytes read
1 block, 81 bytes transmitted
Elapsed time: 00:00:04.765
CPU time:     00:00:00.896
#RJIN *;COMPRESS=NO;TRUNCATE=NO
RJIN Settings - File reference: "*"
  INCODE=ASCII; XPARENT=NO; TRUNCATE=NO; COMPRESS=NO
```

```
  Effective MAXRPB=255
  WAIT=3,0
File characteristics:
  IBMT1,OLD;REC=-80,3,F,ASCII;NOCCTL;DISC=???;ACC=IN
03OCT94-09:43:57.0: Issuing transmit bid
03OCT94-09:44:00.5: Transmit bid acknowledged: starting transfer
#RJEOD
03OCT94-09:44:02.0: Transmit complete                        <-----
1 record, 80 bytes read
1 block, 81 bytes transmitted
Elapsed time: 00:00:05.011
CPU time:     00:00:00.171
#RJOUT $STDLIST
RJOUT Settings - File reference: "$STDLIST"
  OUTCODE=ASCII; WAIT=3,0
  EMPTYOK=NO
  TRUNCATE=NO; CCTL=YES
  REPEAT=NO; INTERRUPT=NO
File characteristics:
  $STDLIST;REC=-80,1,U,ASCII;CCTL;DISC=0;ACC=OUT;NOBUF
03OCT94-09:44:02.3: Waiting for line bid
03OCT94-09:44:12.9: Line bid received: starting transfer
03OCT94-09:44:13.0: ----------Start data set---------------------
WELCOME TO IBM IN EXPEDITE/DIRECT 941003124332
03OCT94-09:44:13.0: ----------End data set-----------------------
#RJIN *;COMPRESS=NO;TRUNCATE=NO
RJIN Settings - File reference: "*"
  INCODE=ASCII; XPARENT=NO; TRUNCATE=NO; COMPRESS=NO
  Effective MAXRPB=255
  WAIT=3,0
File characteristics:
  IBMT1,OLD;REC=-80,3,F,ASCII;NOCCTL;DISC=???;ACC=IN
03OCT94-09:44:15.2: Issuing transmit bid
03OCT94-09:44:21.5: Transmit bid acknowledged: starting transfer
#RJIN EDISEND;COMPRESS=NO;TRUNCATE=NO
03OCT94-09:44:25.4: Transmit complete                        <-----
1 record, 80 bytes read
1 block, 81 bytes transmitted
Elapsed time: 00:00:10.374
CPU time:     00:00:00.260
RJIN Settings - File reference: "EDISEND"
  INCODE=ASCII; XPARENT=NO; TRUNCATE=NO; COMPRESS=NO
  Effective MAXRPB=255
  WAIT=3,0
File characteristics:
  EDISEND,OLD;REC=-80,1,F,ASCII;NOCCTL;DISC=1;ACC=IN
#RJEOD
03OCT94-09:44:26.9: Transmit complete                        <-----
1 record, 80 bytes read
1 block, 81 bytes transmitted
Elapsed time: 00:00:01.469
CPU time:     00:00:00.124
#RJIN *;COMPRESS=NO;TRUNCATE=NO
RJIN Settings - File reference: "*"
```

```
  INCODE=ASCII; XPARENT=NO; TRUNCATE=NO; COMPRESS=NO
  Effective MAXRPB=255
  WAIT=3,0
File characteristics:
  IBMT1,OLD;REC=-80,3,F,ASCII;NOCCTL;DISC=???;ACC=IN
03OCT94-09:44:27.0: Issuing transmit bid
03OCT94-09:44:27.3: Transmit bid acknowledged: starting transfer
#RJEOD
03OCT94-09:44:28.8: Transmit complete                        <-----
1 record, 80 bytes read
1 block, 81 bytes transmitted
Elapsed time: 00:00:01.835
CPU time:     00:00:00.150
#RJOUT EDIRECV
RJOUT Settings - File reference: "EDIRECV"
  OUTCODE=ASCII; WAIT=3,0
  EMPTYOK=NO
  TRUNCATE=NO; CCTL=YES
  REPEAT=NO; INTERRUPT=NO
File characteristics:
  EDIRECV,OLD;REC=-80,16,F,ASCII;DISC=1023;ACC=OUT
03OCT94-09:44:28.9: Waiting for line bid
03OCT94-09:44:31.5: Line bid received: starting transfer
03OCT94-09:44:32.0: Warning: Receiving routed Punch file data
03OCT94-09:44:32.3: Receive complete                         <-----
2 records, 118 bytes written
1 block, 121 bytes received
Elapsed time: 00:00:03.411
CPU time:     00:00:00.120
#RJIN *;COMPRESS=NO;TRUNCATE=NO
RJIN Settings - File reference: "*"
  INCODE=ASCII; XPARENT=NO; TRUNCATE=NO; COMPRESS=NO
  Effective MAXRPB=255
  WAIT=3,0
File characteristics:
  IBMT1,OLD;REC=-80,3,F,ASCII;NOCCTL;DISC=???;ACC=IN
03OCT94-09:44:32.4: Issuing transmit bid
03OCT94-09:44:32.7: Transmit bid acknowledged: starting transfer
#RJEOD
03OCT94-09:44:34.1: Transmit complete                        <-----
1 record, 80 bytes read
1 block, 81 bytes transmitted
Elapsed time: 00:00:01.764
CPU time:     00:00:00.150
#RJOUT $STDLIST
RJOUT Settings - File reference: "$STDLIST"
  OUTCODE=ASCII; WAIT=3,0
  EMPTYOK=NO
  TRUNCATE=NO; CCTL=YES
  REPEAT=NO; INTERRUPT=NO
File characteristics:
  $STDLIST;REC=-80,1,U,ASCII;CCTL;DISC=0;ACC=OUT;NOBUF
03OCT94-09:44:34.4: Waiting for line bid
03OCT94-09:44:34.8: Line bid received: starting transfer
```

```
03OCT94-09:44:34.9: ----------Start data set---------------------
DFH2001I INVALID TRANSACTION IDENTIFICATION RECE -
     PLEASE RESUBMIT  12:43:52
03OCT94-09:44:34.9: ----------End data set-----------------------
#RJOUT $STDLIST
RJOUT Settings - File reference: "$STDLIST"
  OUTCODE=ASCII; WAIT=3,0
  EMPTYOK=NO
  TRUNCATE=NO; CCTL=YES
  REPEAT=NO; INTERRUPT=NO
File characteristics:
  $STDLIST;REC=-80,1,U,ASCII;CCTL;DISC=0;ACC=OUT;NOBUF
03OCT94-09:44:35.7: Waiting for line bid
03OCT94-09:44:36.8: Line bid received: starting transfer
03OCT94-09:44:36.9: ----------Start data set---------------------
DFH2001I INVALID TRANSACTION IDENTIFICATION RECE -
     PLEASE RESUBMIT  16:43:54
03OCT94-09:44:36.9: ----------End data set-----------------------
#RJEND
```

Here are the contents of the **EDIRECV** file:

```
SDIERR  IED110305S  NO IELOGON COMMAND WAS SPECIFIED
```

It appears there are three problems with our first attempt:

- We received an **RVI** (Reverse Interrupt) after sending our first logon.
- We received an Invalid Transaction Identification message from Advantis
- When we examine the contents of the **EDIRECV** file, we find that it contains a message stating that no **IELOGON** had been specified.

When you encounter more than one problem, it is often a good idea to attempt to solve the first problem encountered. Like program compilation errors, it is often the case that a single error can cause multiple subsequent errors to occur.

The first possible problem is the Reverse Interrupt (**RVI**) received after the initial logon. Although this Reverse Interrupt is *not* a fatal error, it does indicate that Advantis has an urgent message to send to us.

To receive this apparently urgent message, we'll add an **#RJOUT** to our script at the point following the command that generated the **RVI** message, as follows:

```
#RJIN *;COMPRESS=NO;TRUNCATE=NO
/*LOGON account,account,password/*SELECT EDIRECT/*USERDATA BSCEDI
#RJEOD
#RJOUT $STDLIST
#RJIN *;COMPRESS=NO;TRUNCATE=NO
IELOGON ACCOUNT(account) USERID(account) PASSWORD(password);
#RJEOD
.
.
.
```

Here are the results of our second attempt:

```
:NERJE IBMT2
Network/S - Telamon, Inc. (C) 1987, 1994 - 03OCT94-12:22:55.3
Version A.03.00nm ...
Licensee: ...
#RJLINE 3780;CONNECT=DIAL,"advantis number",dialer;LINECODE=EBCDIC
RJLINE Settings
  3780; LINECODE=EBCDIC; PRI=NORMAL
  DEV=device,speed; WAIT=0,0
  CONNECT=DIAL,"advantis number",dialer
Network/S: Engine Version: A.6.5 (Dual) (Sync) (AutoDial) (8MHz)
#RJIN *;COMPRESS=NO;TRUNCATE=NO
RJIN Settings - File reference: "*"
  INCODE=ASCII; XPARENT=NO; TRUNCATE=NO; COMPRESS=NO
  Effective MAXRPB=255
  WAIT=3,0
File characteristics:
  IBMT2,OLD;REC=-80,3,F,ASCII;NOCCTL;DISC=???;ACC=IN
03OCT94-12:22:57.5: Waiting for connection
03OCT94-12:23:09.5: On Line (DSR high)
03OCT94-12:23:09.7: Issuing transmit bid
03OCT94-12:23:10.0: Transmit bid acknowledged: starting transfer
#RJEOD
03OCT94-12:23:10.5: Warning: RVI (Reverse Interrupt) Received
03OCT94-12:23:10.6: Transmit complete                              <-----
1 record, 80 bytes read
1 block, 81 bytes transmitted
Elapsed time: 00:00:00.979
CPU time:     00:00:00.144
#RJOUT $STDLIST
RJOUT Settings - File reference: "$STDLIST"
  OUTCODE=ASCII; WAIT=3,0
  EMPTYOK=NO
  TRUNCATE=NO; CCTL=YES
  REPEAT=NO; INTERRUPT=NO
File characteristics:
  $STDLIST;REC=-80,1,U,ASCII;CCTL;DISC=0;ACC=OUT;NOBUF
03OCT94-12:23:12.9: Waiting for line bid
03OCT94-12:23:18.5: Line bid received: starting transfer
03OCT94-12:23:18.6: ----------Start data set---------------------
WELCOME TO IBM IN EXPEDITE/DIRECT 941003152237
03OCT94-12:23:18.7: ----------End data set-----------------------
#RJIN *;COMPRESS=NO;TRUNCATE=NO
RJIN Settings - File reference: "*"
  INCODE=ASCII; XPARENT=NO; TRUNCATE=NO; COMPRESS=NO
  Effective MAXRPB=255
  WAIT=3,0
File characteristics:
  IBMT2,OLD;REC=-80,3,F,ASCII;NOCCTL;DISC=???;ACC=IN
03OCT94-12:23:20.7: Issuing transmit bid
03OCT94-12:23:20.9: Transmit bid acknowledged: starting transfer
#RJEOD
03OCT94-12:23:25.6: Transmit complete                          <-----
```

```
1 record, 80 bytes read
1 block, 81 bytes transmitted
Elapsed time: 00:00:05.069
CPU time:     00:00:00.222
#RJOUT $STDLIST
RJOUT Settings - File reference: "$STDLIST"
  OUTCODE=ASCII; WAIT=3,0
  EMPTYOK=NO
  TRUNCATE=NO; CCTL=YES
  REPEAT=NO; INTERRUPT=NO
File characteristics:
  $STDLIST;REC=-80,1,U,ASCII;CCTL;DISC=0;ACC=OUT;NOBUF
03OCT94-12:23:26.0: Waiting for line bid
03OCT94-12:23:28.2: Line bid received: starting transfer
03OCT94-12:23:28.6: ----------Start data set---------------------
EXPEDITE/DIRECT RECEIVE INPUT DATA SET 0001
03OCT94-12:23:28.7: ----------End data set-----------------------
#RJIN *;COMPRESS=NO;TRUNCATE=NO
RJIN Settings - File reference: "*"
  INCODE=ASCII; XPARENT=NO; TRUNCATE=NO; COMPRESS=NO
  Effective MAXRPB=255
  WAIT=3,0
File characteristics:
  IBMT2,OLD;REC=-80,3,F,ASCII;NOCCTL;DISC=???;ACC=IN
03OCT94-12:25:54.3: Issuing transmit bid
03OCT94-12:25:54.5: Transmit bid acknowledged: starting transfer
#RJIN EDISEND;COMPRESS=NO;TRUNCATE=NO
03OCT94-12:25:55.1: Transmit complete                        <-----
1 record, 80 bytes read
1 block, 81 bytes transmitted
Elapsed time: 00:00:00.972
CPU time:     00:00:00.128
RJIN Settings - File reference: "EDISEND"
  INCODE=ASCII; XPARENT=NO; TRUNCATE=NO; COMPRESS=NO
  Effective MAXRPB=255
  WAIT=3,0
File characteristics:
  EDISEND,OLD;REC=-80,1,F,ASCII;NOCCTL;DISC=1;ACC=IN
#RJEOD
03OCT94-12:25:56.6: Transmit complete                        <-----
1 record, 80 bytes read
1 block, 81 bytes transmitted
Elapsed time: 00:00:01.490
CPU time:     00:00:00.135
#RJIN *;COMPRESS=NO;TRUNCATE=NO
RJIN Settings - File reference: "*"
  INCODE=ASCII; XPARENT=NO; TRUNCATE=NO; COMPRESS=NO
  Effective MAXRPB=255
  WAIT=3,0
File characteristics:
  IBMT2,OLD;REC=-80,3,F,ASCII;NOCCTL;DISC=???;ACC=IN
03OCT94-12:25:56.8: Issuing transmit bid
03OCT94-12:25:57.0: Transmit bid acknowledged: starting transfer
#RJEOD
```

```
03OCT94-12:25:58.5: Transmit complete                          <-----
1 record, 80 bytes read
1 block, 81 bytes transmitted
Elapsed time: 00:00:01.820
CPU time:     00:00:00.150
#RJOUT EDIRECV
RJOUT Settings - File reference: "EDIRECV"
  OUTCODE=ASCII; WAIT=3,0
  EMPTYOK=NO
  TRUNCATE=NO; CCTL=YES
  REPEAT=NO; INTERRUPT=NO
File characteristics:
  EDIRECV,OLD;REC=-80,16,F,ASCII;DISC=1023;ACC=OUT
03OCT94-12:25:58.6: Waiting for line bid
03OCT94-12:26:01.5: Line bid received: starting transfer
03OCT94-12:26:01.8: Warning: Receiving routed Punch file data
03OCT94-12:26:02.2: Receive complete                           <-----
1 record, 43 bytes written
1 block, 45 bytes received
Elapsed time: 00:00:03.563
CPU time:     00:00:00.118
#RJIN *;COMPRESS=NO;TRUNCATE=NO
RJIN Settings - File reference: "*"
  INCODE=ASCII; XPARENT=NO; TRUNCATE=NO; COMPRESS=NO
  Effective MAXRPB=255
  WAIT=3,0
File characteristics:
  IBMT2,OLD;REC=-80,3,F,ASCII;NOCCTL;DISC=???;ACC=IN
03OCT94-12:26:02.3: Issuing transmit bid
03OCT94-12:26:02.5: Transmit bid acknowledged: starting transfer
#RJEOD
03OCT94-12:26:04.1: Transmit complete                          <-----
1 record, 80 bytes read
1 block, 81 bytes transmitted
Elapsed time: 00:00:01.922
CPU time:     00:00:00.151
#RJOUT $STDLIST
RJOUT Settings - File reference: "$STDLIST"
  OUTCODE=ASCII; WAIT=3,0
  EMPTYOK=NO
  TRUNCATE=NO; CCTL=YES
  REPEAT=NO; INTERRUPT=NO
File characteristics:
  $STDLIST;REC=-80,1,U,ASCII;CCTL;DISC=0;ACC=OUT;NOBUF
03OCT94-12:26:04.4: Waiting for line bid
03OCT94-12:26:07.0: Line bid received: starting transfer
03OCT94-12:26:07.2: ----------Start data set---------------------
EXPEDITE/DIRECT RECEIVE INPUT DATA SET 0003
03OCT94-12:26:07.4: ----------Start data set---------------------
#RJOUT $STDLIST
RJOUT Settings - File reference: "$STDLIST"
  OUTCODE=ASCII; WAIT=3,0
  EMPTYOK=NO
  TRUNCATE=NO; CCTL=YES
```

```
  REPEAT=NO; INTERRUPT=NO
File characteristics:
  $STDLIST;REC=-80,1,U,ASCII;CCTL;DISC=0;ACC=OUT;NOBUF
03OCT94-12:26:07.9: Waiting for line bid
03OCT94-12:26:08.7: Line bid received: starting transfer
03OCT94-12:26:09.0: ----------Start data set---------------------
EXPEDITE/DIRECT -- SESSION END
03OCT94-12:26:09.5: ----------End data set-----------------------
#RJEND
```

When we examine the **EDIRECV** file we find:

```
EXPEDITE/DIRECT RECEIVE INPUT DATA SET 0002
```

Notice that there are 2 different problems now:

- Advantis is still sending us the Reverse Interrupt (**RVI**) message.
- Advantis is sending us acknowledgments that they received what we are sending but we are NOT receiving the data or the EDILOG.

The **RVI** message is, again, a non-fatal error and we've placed an **#RJOUT** at this point to receive the message. The "urgent" message turned out to be their "Welcome" message! Because we placed an **#RJOUT** at this point and successfully received this message, we can safely disregard the **RVI** message, which is displayed because Network/S does *not* look ahead to determine that the next command after receiving the **RVI** is an **#RJOUT**.

The second problem seems to be caused by Advantis sending us messages to acknowledge receiving what we have sent. For example, notice the message received when we performed an **#RJOUT** after we sent the **IEL-OGON**:

```
EXPEDITE/DIRECT RECEIVE INPUT DATA SET 0001
```

By inserting **#RJOUT** commands after each **#RJIN** to Advantis, we may be able to capture the "Expedite/Direct Receive" messages as well as the expected data files**.**

After inserting the **#RJOUT**s, the script appears as follows:

```
#RJCOMMENT Define the communications line with the #RJLINE command
#RJCOMMENT I am using the SADL autodialer parameter because I am
#RJCOMMENT dialing with the Racal Milgo 4850PA modem. I am using a
#RJCOMMENT LINECODE of EBCDIC because I understand that most
#RJCOMMENT parties use this LINECODE (note that the LINECODE is
#RJCOMMENT not to be confused with the format of the data I am
#RJCOMMENT sending and receiving from Advantis.
#RJLINE 3780;CONNECT=DIAL,"advantis number",SADL;LINECODE=EBCDIC
#RJCOMMENT Send the logons using the #RJIN commands. I placed a
#RJCOMMENT #RJEOD command after the second #RJIN because I am
#RJCOMMENT done sending data (the logons)
#RJIN *;COMPRESS=NO;TRUNCATE=NO
/*LOGON account,account,password/*SELECT EDIRECT/*USERDATA BSCEDI
#RJEOD
#RJCOMMENT Inserted an #RJOUT here because we receive an RVI
#RJOUT $STDLIST
```

```
#RJIN *;COMPRESS=NO;TRUNCATE=NO
IELOGON ACCOUNT(account) USERID(account) PASSWORD(password);
#RJEOD
#RJCOMMENT I placed an #RJOUT here because I think Advantis might
#RJCOMMENT be sending me a welcome message. I was wrong; Advantis
#RJCOMMENT sends a message but it is NOT their welcome message. The
#RJCOMMENT message they send merely indicates that they received my
#RJCOMMENT IELOGON.
#RJOUT $STDLIST
#RJCOMMENT Transmit the SEND command to Advantis followed
#RJCOMMENT by the data
#RJIN *;COMPRESS=NO;TRUNCATE=NO
SEND ACCOUNT(account) USERID(account);
#RJIN EDISEND;COMPRESS=NO;TRUNCATE=NO
#RJEOD
#RJCOMMENT Test 3 - Inserted #RJOUT here to capture Expedite/Direct
#RJCOMMENT "message received" message from Advantis
#RJOUT $STDLIST
#RJCOMMENT Transmit the RECEIVE command to Advantis
#RJIN *;COMPRESS=NO;TRUNCATE=NO
RECEIVE CLASS(X12);
#RJEOD
#RJCOMMENT Wait for "message received"
#RJOUT $STDLIST
#RJCOMMENT Receive the contents of the mailbox into EDIRECV
#RJOUT EDIRECV
#RJCOMMENT Transmit log (and logoff) request
#RJIN *;COMPRESS=NO;TRUNCATE=NO
RECEIVE CLASS(EDILOG);
#RJEOD
#RJCOMMENT Wait for "message received"
#RJOUT $STDLIST
#RJCOMMENT Wait for EDILOG data
#RJOUT $STDLIST
#RJCOMMENT Wait for logoff confirmation
#RJOUT $STDLIST
#RJEND
```

The results of our third test (comments removed) are as follows:

```
:NERJE IBMT3
Network/S - Telamon, Inc. (C)...
Version A.03.00...
Licensee: ...
#RJLINE 3780;CONNECT=DIAL,"advantis number",SADL;LINECODE=EBCDIC
RJLINE Settings
  3780; LINECODE=EBCDIC; PRI=NORMAL
  DEV=device,speed; WAIT=0,0
  CONNECT=DIAL,"advantis number",SADL
Network/S: Engine Version: A.6.5 (Dual) (Sync) (AutoDial) (8MHz)
#RJIN *;COMPRESS=NO;TRUNCATE=NO
RJIN Settings - File reference: "*"
  INCODE=ASCII; XPARENT=NO; TRUNCATE=NO; COMPRESS=NO
  Effective MAXRPB=255
```

```
  WAIT=3,0
File characteristics:
  IBMT3,OLD;REC=-80,3,F,ASCII;NOCCTL;DISC=???;ACC=IN
04OCT94-14:38:48.8: Waiting for connection
04OCT94-14:38:59.8: On Line (DSR high)
04OCT94-14:39:00.0: Issuing transmit bid
04OCT94-14:39:00.2: Transmit bid acknowledged: starting transfer
#RJEOD
04OCT94-14:39:00.8: Warning: RVI (Reverse Interrupt) Received
04OCT94-14:39:00.8: Transmit complete                          <-----
1 record, 80 bytes read
1 block, 81 bytes transmitted
Elapsed time: 00:00:00.971
CPU time:     00:00:00.140
#RJOUT $STDLIST
RJOUT Settings - File reference: "$STDLIST"
  OUTCODE=ASCII; WAIT=3,0
  EMPTYOK=NO
  TRUNCATE=NO; CCTL=YES
  REPEAT=NO; INTERRUPT=NO
File characteristics:
  $STDLIST;REC=-80,1,U,ASCII;CCTL;DISC=0;ACC=OUT;NOBUF
04OCT94-14:39:03.1: Waiting for line bid
04OCT94-14:39:09.1: Line bid received: starting transfer
04OCT94-14:39:09.2: ----------Start data set---------------------
WELCOME TO IBM IN EXPEDITE/DIRECT 941004173825
04OCT94-14:39:09.3: ----------End data set-----------------------
#RJIN *;COMPRESS=NO;TRUNCATE=NO
RJIN Settings - File reference: "*"
  INCODE=ASCII; XPARENT=NO; TRUNCATE=NO; COMPRESS=NO
  Effective MAXRPB=255
  WAIT=3,0
File characteristics:
  IBMT3,OLD;REC=-80,3,F,ASCII;NOCCTL;DISC=???;ACC=IN
04OCT94-14:39:19.2: Issuing transmit bid
04OCT94-14:39:19.4: Transmit bid acknowledged: starting transfer
#RJEOD
04OCT94-14:39:20.8: Transmit complete                          <-----
1 record, 80 bytes read
1 block, 81 bytes transmitted
Elapsed time: 00:00:01.779
CPU time:     00:00:00.150
#RJOUT $STDLIST
RJOUT Settings - File reference: "$STDLIST"
  OUTCODE=ASCII; WAIT=3,0
  EMPTYOK=NO
  TRUNCATE=NO; CCTL=YES
  REPEAT=NO; INTERRUPT=NO
File characteristics:
  $STDLIST;REC=-80,1,U,ASCII;CCTL;DISC=0;ACC=OUT;NOBUF
04OCT94-14:39:21.2: Waiting for line bid
04OCT94-14:39:21.9: Line bid received: starting transfer
04OCT94-14:39:22.0: ----------Start data set---------------------
EXPEDITE/DIRECT RECEIVE INPUT DATA SET 0001
```

```
04OCT94-14:39:22.2: ----------End data set----------------------
#RJIN *;COMPRESS=NO;TRUNCATE=NO
RJIN Settings - File reference: "*"
  INCODE=ASCII; XPARENT=NO; TRUNCATE=NO; COMPRESS=NO
  Effective MAXRPB=255
  WAIT=3,0
File characteristics:
  IBMT3,OLD;REC=-80,3,F,ASCII;NOCCTL;DISC=???;ACC=IN
04OCT94-14:39:22.7: Issuing transmit bid
04OCT94-14:39:23.0: Transmit bid acknowledged: starting transfer
#RJIN EDISEND;COMPRESS=NO;TRUNCATE=NO
04OCT94-14:39:23.6: Transmit complete                        <-----
1 record, 80 bytes read
1 block, 81 bytes transmitted
Elapsed time: 00:00:00.972
CPU time:     00:00:00.124
RJIN Settings - File reference: "EDISEND"
  INCODE=ASCII; XPARENT=NO; TRUNCATE=NO; COMPRESS=NO
  Effective MAXRPB=255
  WAIT=3,0
File characteristics:
  EDISEND,OLD;REC=-80,1,F,ASCII;NOCCTL;DISC=1;ACC=IN
#RJEOD
04OCT94-14:39:28.4: Transmit complete                        <-----
1 record, 80 bytes read
1 block, 81 bytes transmitted
Elapsed time: 00:00:04.776
CPU time:     00:00:00.195
#RJOUT $STDLIST
RJOUT Settings - File reference: "$STDLIST"
  OUTCODE=ASCII; WAIT=3,0
  EMPTYOK=NO
  TRUNCATE=NO; CCTL=YES
  REPEAT=NO; INTERRUPT=NO
File characteristics:
  $STDLIST;REC=-80,1,U,ASCII;CCTL;DISC=0;ACC=OUT;NOBUF
04OCT94-14:39:28.7: Waiting for line bid
04OCT94-14:39:34.3: Line bid received: starting transfer
04OCT94-14:39:34.4: ----------Start data set---------------------
EXPEDITE/DIRECT RECEIVE INPUT DATA SET 0002
04OCT94-14:39:34.5: ----------End data set----------------------
#RJIN *;COMPRESS=NO;TRUNCATE=NO
RJIN Settings - File reference: "*"
  INCODE=ASCII; XPARENT=NO; TRUNCATE=NO; COMPRESS=NO
  Effective MAXRPB=255
  WAIT=3,0
File characteristics:
  IBMT3,OLD;REC=-80,3,F,ASCII;NOCCTL;DISC=???;ACC=IN
04OCT94-14:39:35.1: Issuing transmit bid
04OCT94-14:39:35.4: Transmit bid acknowledged: starting transfer
#RJEOD
04OCT94-14:39:36.8: Transmit complete                        <-----
1 record, 80 bytes read
1 block, 81 bytes transmitted
```

```
Elapsed time: 00:00:01.780
CPU time:     00:00:00.151
#RJOUT $STDLIST
RJOUT Settings - File reference: "$STDLIST"
  OUTCODE=ASCII; WAIT=3,0
  EMPTYOK=NO
  TRUNCATE=NO; CCTL=YES
  REPEAT=NO; INTERRUPT=NO
File characteristics:
  $STDLIST;REC=-80,1,U,ASCII;CCTL;DISC=0;ACC=OUT;NOBUF
04OCT94-14:39:37.1: Waiting for line bid
04OCT94-14:39:37.3: Line bid received: starting transfer
04OCT94-14:39:37.3: ----------Start data set---------------------
EXPEDITE/DIRECT RECEIVE INPUT DATA SET 0003
04OCT94-14:39:37.4: ----------End data set-----------------------
#RJOUT EDIRECV
RJOUT Settings - File reference: "EDIRECV"
  OUTCODE=ASCII; WAIT=3,0
  EMPTYOK=NO
  TRUNCATE=NO; CCTL=YES
  REPEAT=NO; INTERRUPT=NO
File characteristics:
  EDIRECV,OLD;REC=-80,16,F,ASCII;DISC=1023;ACC=OUT
04OCT94-14:39:38.1: Waiting for line bid
04OCT94-14:39:39.3: Line bid received: starting transfer
04OCT94-14:39:39.7: Warning: Receiving routed Punch file data
04OCT94-14:39:40.1: Receive complete                          <-----
1 record, 80 bytes written
1 block, 82 bytes received
Elapsed time: 00:00:01.964
CPU time:     00:00:00.100
#RJIN *;COMPRESS=NO;TRUNCATE=NO
RJIN Settings - File reference: "*"
  INCODE=ASCII; XPARENT=NO; TRUNCATE=NO; COMPRESS=NO
  Effective MAXRPB=255
  WAIT=3,0
File characteristics:
  IBMT3,OLD;REC=-80,3,F,ASCII;NOCCTL;DISC=???;ACC=IN
04OCT94-14:39:40.2: Issuing transmit bid
04OCT94-14:39:40.4: Transmit bid acknowledged: starting transfer
#RJEOD
04OCT94-14:39:42.0: Transmit complete                         <-----
1 record, 80 bytes read
1 block, 81 bytes transmitted
Elapsed time: 00:00:01.912
CPU time:     00:00:00.148
#RJOUT $STDLIST
RJOUT Settings - File reference: "$STDLIST"
  OUTCODE=ASCII; WAIT=3,0
  EMPTYOK=NO
  TRUNCATE=NO; CCTL=YES
  REPEAT=NO; INTERRUPT=NO
File characteristics:
  $STDLIST;REC=-80,1,U,ASCII;CCTL;DISC=0;ACC=OUT;NOBUF
```

```
04OCT94-14:39:42.3: Waiting for line bid
04OCT94-14:39:45.6: Line bid received: starting transfer
04OCT94-14:39:45.7: ----------Start data set--------------------
EXPEDITE/DIRECT RECEIVE INPUT DATA SET 0004
04OCT94-14:39:45.8: ----------End data set----------------------
#RJOUT $STDLIST
RJOUT Settings - File reference: "$STDLIST"
  OUTCODE=ASCII; WAIT=3,0
  EMPTYOK=NO
  TRUNCATE=NO; CCTL=YES
  REPEAT=NO; INTERRUPT=NO
File characteristics:
  $STDLIST;REC=-80,1,U,ASCII;CCTL;DISC=0;ACC=OUT;NOBUF
04OCT94-14:39:46.5: Waiting for line bid
04OCT94-14:39:47.6: Line bid received: starting transfer
04OCT94-14:39:47.7: ----------Start data set--------------------
*** EXPEDITE/DIRECT SESSION LOG ***
EXPEDITE/DIRECT SESSION (    ) STARTED AT 17:38:37
IELOGON ACCOUNT(account) USERID(account)
SEND FREE FMT: CHARS(0000056) CLASS(X12001) TO ACCT(account)....
RECEIVE MESSAGES: CLASS(#E2) FROM ALL USERS
EXPEDITE/DIRECT SESSION ENDED AT 17:39:02 RESPONSE(00000)
SEND TOTALS: X12(0001) UCS(0000) EDIFACT(0000) UN/TDI(0000) FFMT(0001)
RECV TOTALS: X12(0001) UCS(0000) EDIFACT(0000) UN/TDI(0000) FFMT(0000)
04OCT94-14:39:48.9: ----------End data set----------------------
#RJOUT $STDLIST
RJOUT Settings - File reference: "$STDLIST"
  OUTCODE=ASCII; WAIT=3,0
  EMPTYOK=NO
  TRUNCATE=NO; CCTL=YES
  REPEAT=NO; INTERRUPT=NO
File characteristics:
  $STDLIST;REC=-80,1,U,ASCII;CCTL;DISC=0;ACC=OUT;NOBUF
04OCT94-14:39:50.4: Waiting for line bid
04OCT94-14:39:51.3: Line bid received: starting transfer
04OCT94-14:39:51.4: ----------Start data set--------------------
EXPEDITE/DIRECT -- SESSION END
04OCT94-14:39:51.5: ----------End data set----------------------
#RJEND
```

It worked!

You can see that the order in which things are sent and received *must* be in lock-step with the remote system's sequence of operations. If you are trying to write a script and you don't know if the other party is trying to send you data after you have sent them data, you can insert an #**RJOUT** after each #**RJIN** with the following parameters:

```
#RJOUT filename;TIMEOUTOK=YES;WAIT=0,10
```

These parameters combined will allow you to receive a file if one is sent, or continue if no file is sent. The **WAIT** value may have to be adjusted depending on the remote system. If the wait time is too short the remote system may not get a chance to bid for the line; if it is too long, the remote system may timeout and drop the line.

Notice that the output from the above command sequence, with **VERBOSE** and **SHOW** set to **NO**, appears as:

```
#RJLINE 3780;CONNECT=DIAL,"advantis number",SADL;LINECODE=EBCDIC; &
#  VERBOSE=NO;SHOW=NO
#RJIN *;COMPRESS=NO;TRUNCATE=NO
#RJEOD
#RJOUT $STDLIST
WELCOME TO IBM IN EXPEDITE/DIRECT 941004173825
#RJIN *;COMPRESS=NO;TRUNCATE=NO
#RJEOD
#RJOUT $STDLIST
EXPEDITE/DIRECT RECEIVE INPUT DATA SET 0001
#RJIN *;COMPRESS=NO;TRUNCATE=NO
#RJIN EDISEND;COMPRESS=NO;TRUNCATE=NO
#RJEOD
#RJOUT $STDLIST
EXPEDITE/DIRECT RECEIVE INPUT DATA SET 0002
#RJIN *;COMPRESS=NO;TRUNCATE=NO
#RJEOD
#RJOUT $STDLIST
EXPEDITE/DIRECT RECEIVE INPUT DATA SET 0003
#RJOUT EDIRECV
#RJIN *;COMPRESS=NO;TRUNCATE=NO
#RJEOD
#RJOUT $STDLIST
EXPEDITE/DIRECT RECEIVE INPUT DATA SET 0004
#RJOUT $STDLIST
*** EXPEDITE/DIRECT SESSION LOG ***
EXPEDITE/DIRECT SESSION (    ) STARTED AT 17:38:37
IELOGON ACCOUNT(account) USERID(account)
SEND FREE FMT: CHARS(0000056) CLASS(X12001) TO ACCT(account)....
RECEIVE MESSAGES: CLASS(#E2) FROM ALL USERS
EXPEDITE/DIRECT SESSION ENDED AT 17:39:02 RESPONSE(00000)
SEND TOTALS: X12(0001) UCS(0000) EDIFACT(0000) UN/TDI(0000) FFMT(0001)
RECV TOTALS: X12(0001) UCS(0000) EDIFACT(0000) UN/TDI(0000) FFMT(0000)
#RJOUT $STDLIST
EXPEDITE/DIRECT -- SESSION END
#RJEND
```

While this is a substantially smaller amount of output, it also reduces the amount of information available to monitor the job's progress.

CHAPTER 3

Commands

List of Network/S Commands

Note: In all of the following commands, any option that can be stated as **OPT=YES** may be abbreviated as **OPT**.

HELP

Access QHELP sub-system

The **#HELP**, or **#RJHELP**, command gives you access to the QHELP sub-system, provided complements of Robelle Consulting, Ltd. Entering **#HELP *command*** will display information about that specified Network/S command. Entering **#HELP** alone will display general information about the Network/S system and allow you to get more detailed descriptions of various subjects. Enter a "?" at the QHELP prompt to get information about the operation of the QHELP sub-system itself.

For example, to get help about the **#RJLINE** command, enter:

```
#HELP RJLINE
```

To enter the help sub-system for browsing, enter:

```
#HELP
```

To get help about a particular subject in a Help topic list, enter all of the necessary sub-headings to get to the subject. For instance, to see the modem configuration information, enter:

```
#HELP CONFIGURATION,MODEM
```

To generate a hard copy listing about a particular item, enter:

```
#PHELP item
```

The formal designator for the print file is **QHELPOUT**.

To generate a hard copy listing of the entire Help file, enter:

```
#PHELP @
```

The help file used is **HELP.NETWORKS.TELAMON**.

REDO

Edit and Re-execute Previous Line

The **#REDO** command provides a means to edit and repeat the last line entered. It functions in much the same fashion as the MPE **:REDO** command with some additional capabilities. The editing commands available are:

I	Insert text at current location
R	Replace text
D	Delete text
X	Extend text (add at end of line)
U	Undo last change
UU	Undo all changes
?	Help

Any other character automatically replaces the character above it.

Example

```
#RJLINE 3780;LINECOE=ASCII
Parameter Error: Invalid option
RJLINE 3780;LINECOE=ASCII
           ^^^^^^^
Invalid or Unknown command or option
#REDO
Alter(?): RJLINE 3780;LINECOE=ASCII
          ________________ID
Alter(?): RJLINE 3780;LINECDOE=ASCII
          U
Alter(?): RJLINE 3780;LINECOE=ASCII
          ________________ID
Alter(?): RJLINE 3780;LINECODE=ASCII
          ____________________REBCDIC
Alter(?): RJLINE 3780;LINECODE=EBCDIC
          cr
RJLINE Settings
  3780; LINECODE=EBCDIC...
```

REDO will execute Telamon's Alter procedure, if you have it installed on your system.

DO - Re-execute previous line

The **#DO** command provides a means to repeat the last line entered. It functions like **#REDO** with no editing.

RJABORT

Terminate with an Error Condition

The #**RJABORT** command forces Network/S to terminate in an error state, setting the system JCW and the **NETWORK** JCW to -1. Use this command to for Network/S to terminate with an error condition when using the #**RJCONTINUE** command to inhibit normal error processing.

Syntax

#**RJABORT**

RJBID

Initiate a Line Bid Sequence

The **#RJBID** command is used to cause the Engine to initiate a bid sequence to the remote host. This sequence is identical to that employed by the **#RJIN** command, but no data is transferred.

#RJBID should usually be used when one or both of the **ID/REMID** parameters have been specified in the **#RJLINE** command and you want ID validation to occur before any data is transferred. It should also be used in cases where you are expected to be able to send a Bid-EOT sequence, such as when communicating with the Advantis network.

If an **#RJBID** precedes an **#RJIN/RJIO**, the bid will not be repeated in the **#RJIN/RJIO** command. If an **#RJBID** precedes an **#RJOUT**, **#RJLIST**, **#RJPUNCH** or **#RJEOD**, an **EOT** will be sent first, returning the line to the control state.

JCW, Variable and Indirect File substitution will be performed on the text of the command prior to its evaluation. JCWs and MPE/iX variables are identified by a "!" prefix, Variables are identified by a "%" prefix and Indirect Files are identified by a "^" prefix. When an Indirect File is specified, the first record of the file will be read and substituted for the filename in the command. For example, to substitute the variables "M" and "S" in the **WAIT** option, enter:

```
#RJCALC %M = 2
#RJCALC %S = 30
#RJBID;WAIT=%M,%S
```

This evaluates to:

```
#RJBID;WAIT=2,30
```

Syntax:

```
#RJBID [;WAIT=[minutes] [,seconds] ]
```

RJBOF/RJEOF

Copy $STDINX to Temp file

The commands **#RJIF**, **#RJGOTO** and **#RJLABEL** are not permitted when read from **$STDINX**. This is enforced because of the necessity for rewinding and re-reading the command file when performing branch operations, actions which are not possible with the **$STDINX** file.

One solution to this problem is to put the commands into a file and to have Network/S read the commands from that file. The other solution is to use the **#RJBOF** command. When Network/S encounters this command, a temporary file is created and the remainder of the commands found in **$STDINX** are copied to this file, up to, but not including the terminating command, **#RJEOF**. This temporary file is then used as the source of Network/S commands.

This command is valid only when it has been read from **$STDINX**.

Take care *not* to place any Network/S commands following the **#RJEOF** command as they will be ignored by Network/S.

Syntax

```
#RJBOF
Network/S commands
#RJEOF
```

Example

```
!JOB RJE,OPERATOR.SYS
!SETJCW CIERROR = 0
!SETJCW JCW = 0
!CONTINUE
!RUN NETWORK.NETWORKS.TELAMON
#RJBOF
#RJCONTINUE ERROR
#RJLINE 3780;CONNECT=DIAL,5551234
#RJIN MYFILE
#RJEOD
#RJOUT NEWFILE
#RJEND
#RJLABEL ERROR
#RJDISPLAY ^[&dBError!^[&d@
#RJDISPLAY ERRORTYPE = !ERRORTYPE
#RJDISPLAY ERRORCLASS = !ERRORCLASS
#RJDISPLAY ERRORNUM = !ERRORNUM
#RJABORT
#RJEOF
!IF JCW <> 0 OR CIERROR <> 0 THEN
!  TELLOP ****RJE failed****
!ELSE
!  TELLOP ----RJE Complete----
```

```
!ENDIF
!EOJ
```

RJCALC

Modify JCWs and/or Variables

The **#RJCALC** command is used to change the value of a job control word (JCW) or a Network/S variable.

The 26 Network/S variables are available, identified by the letters **A** through **Z**, prefixed with a "%". JCWs and MPE/iX variables are identified by a "!" prefix. For example:

```
#RJCALC !MYJCW = 1
```

sets the JCW called "MYJCW" to 1 and:

```
#RJCALC %D = 10
```

sets the Network/S variable "D" to 10.

Note that only numeric MPE/iX variables may be specified as the target of an **#RJCALC** command. To set string type variables, use **#RJMPE** or **#RJSYS**.

Syntax:

```
#RJCALC {%var|!jcw} = {%var|!jcw|value} [ {+|-} {%var|!jcw|value} ]
```

RJCMDFILE

Switch Command Files

The **#RJCMDFILE** is used to make Network/S start reading its commands from a new source, either a disc file or the default **RJECOM** file (**$STDINX**), if no file is specified.

If no filename is specified, Network/S will revert to reading commands from **RJECOM.**

When End-of-File is encountered on the command file, if Network/S has been run interactively, Network/S will revert to reading commands from **RJECOM**. When encountered in a batch job, End-of-File will terminate Network/S.

Note that the **#RJCMDFILE** command does not provide for nested execution of command files - when the command is executed, the current command file is closed (unless it is **RJECOM**) and control is transferred to the new file.

JCW, Variable and Indirect File substitution will be performed on the text of the command prior to its evaluation. JCWs and MPE/iX variables are identified by a "!" prefix, Variables are identified by a "%" prefix and Indirect Files are identified by a "^" prefix. When an Indirect File is specified, the first record of the file will be read and substituted for the filename in the command. For example, to substitute the variable "N" in the filename, enter:

```
#RJCALC %N = 200
#RJCMDFILE FILE%N
```

This evaluates to:

```
#RJCMDFILE FILE200
```

Syntax:

```
#RJCMDFILE [filename]
```

RJCONTINUE

Specify Error Handler

The **#RJCONTINUE** command provides the means to perform error checking and recovery during the execution of Network/S. With the addition of some extensions, this command duplicates the behavior of the **#RJCONTINUE** command used in HP's RJE product. The parameters passed, and their values, are as defined by HP. Telamon has made various extensions to the **#RJCONTINUE** specification, but the two commands essentially behave identically.

The original usage of the **#RJCONTINUE** command was to either enable or disable access to a pre-compiled error handling procedure, located in a Library file. For the Compatibility Mode version of Network/S, this **#RJCONTINUE** procedure must be located in a Segmented Library (SL) file. For the Native Mode version, the procedure must be located in an XL file.

When an error of some sort occurs, Network/S normally prints an appropriate error message and either terminates if it has been run from a batch job, or terminates the current command file if it has been run from an interactive session. If a branch label is specified with the **#RJCONTINUE** command, Network/S jumps to that location in the current Command File and resumes execution. There, **#RJIF** commands may be used to determine the type and severity of the error that just occurred and can then take corrective action. If a compiled **#RJCONTINUE** procedure has been specified, on the other hand, the procedure is called with the following parameters:

- the file number of the current command file, passed by reference as a 16-bit integer
- a three-word array of error codes identifying the specific error context, passed by reference as a 16-bit integer array
- the command executing at the time of the error, passed by reference as a carriage-return terminated byte array

The procedure returns two parameters to Network/S:

- A new command file to be executed (optional), passed by reference as a blank-terminated byte array
- An action code which informs Network/S what to do next, passed by reference as a 16-bit integer

When specifying an **#RJCONTINUE** procedure, the syntax of the **#RJCONTINUE** command is:

```
#RJCONTINUE @procedurename [ ( lib ) ]
```

Network/S will attempt to load the specified procedure, in the exact specified case, from the SL or XL file specified by *lib.*

When running Network/S on an MPE/V based HP 3000 or in Compatibility Mode on an MPE/XL or MPE/iX based HP 3000, *Lib* can be one of the following:

Lib	SL Search sequence	
---	---	
G	SL.*logongroup*.*logonaccount*	then
	SL.PUB.*logonaccount*	then
	SL.PUB.SYS	

P	SL.PUB.*logonaccount*	then
	SL.PUB.SYS	
S	SL.PUB.SYS only	
GX	SL.*appgroup*.*appaccount*	then
	SL.PUB.*appaccount*	then
	SL.PUB.SYS	
PX	SL.PUB.*appaccount*	then
	SL.PUB.SYS	

where *logongroup* and *logonaccount* refer to the user's logon group and account and *appgroup* and *appaccount* refer to the location where the Network/S application resides, by default, NETWORKS and TELAMON respectively. Note that if Network/S has been placed in PUB.SYS, the only meaningful options for *Lib* are G, P and S since GX and PX would both refer to PUB.SYS.

When running the Native Mode version of Network/S on an MPE/iX based HP 3000, Network/S will attempt to load the specified procedure from the XL file identified by *lib*, which must be a valid filename.

See Appendix D, *Procedure Examples*, for an example **#RJCONTINUE** procedure and an explanation of the parameters passed to the procedure.

When specifying a branch label, the syntax of the **#RJCONTINUE** command is:

```
#RJCONTINUE label | ABORT | NOABORT
```

Label, if specified, must start with an alphabetic character and contain no more than 16 alpha-numeric characters.

ABORT specifies the default behavior, which is to terminate the current operation in the event of an error.

NOABORT specifies that Network/S continue with the next command in the event of an error. A subsequent **#RJIF** command should then be executed to test for an error.

label specifies a branch label elsewhere in the current command file that Network/S should search for and resume execution after, when and if an error occurs.

RJDISC

Send Shutdown Sequence and Disconnect

The **#RJDISC** command will cause Network/S to shutdown and release the communications link without terminating Network/S. For dial-up connections, a **DLE-EOT** sequence will be sent prior to the shutdown; for direct connections an **EOT** will be sent.

Note: Unlike the **#RJEND** command, **#RJDISC** will *not* cause an **EOT** to be sent, if one is pending following an **#RJIN**. If the receiving system expects an **EOT** before line-disconnect, perform an **#RJEOD** before the **#RJDISC**.

Syntax:

```
#RJDISC
```

RJDISPLAY

Display Text

The **#RJDISPLAY** command is used to display messages on **$STDLIST**. The messages can be a mixture of text, JCWs and Network/S variables. JCWs and MPE/iX variables are identified by a "!" prefix and Variables are identified by a "%" prefix. For example, to display the value of the Network/S variable 'A' and the JCW 'CSERROR', you could enter:

```
#RJDISPLAY A's value is %A and CSERROR is !CSERROR.
```

Control characters may be specified by prefixing a character with a "^". For example, to display a message in inverse video, enter:

```
#RJDISPLAY ***CS Error: ^[&dB!ERRORNUM^[&d@ ***
```

The ASCII Escape code (**ESC**) can be entered from the keyboard by typing Control-[- thus the ^[, above, refers to **ESC**.

Syntax:

```
#RJDISPLAY text
```

RJEND

Terminate Network/S

The **#RJEND** command terminates the execution of Network/S. If the **#RJEND** command has been preceded by an **#RJIN** or **#RJBID**, Network/S will execute the equivalent of the **#RJEOD** command before terminating.

Syntax:

```
#RJEND
```

RJEOD

Send End-of-Transmission (EOT)

The **#RJEOD** command forces Network/S to send the End-of-Transmission code (**EOT**) to the remote host system, usually following the execution of an **#RJIN** command. Note that, by default, Network/S will automatically send an **EOT** following an **#RJIN** command when the **#RJIN** command is followed by an **#RJOUT**, **#RJLIST**, **#RJPUNCH** or **#RJEND**.

Syntax:

#**RJEOD**

RJGOTO

Branch within a Command File

The **#RJGOTO** command is used to branch to locations in the command file.

The label must begin with an alphabetic character and may consist of no more than 16 alpha-numeric characters and must be defined using **#RJLABEL**.

Syntax:

#**RJGOTO** ***label***

RJIF

Test JCWs and Branch within a Command File

The **#RJIF** command is used to test the values of job control words (JCW) and/or Network/S variables and to branch to locations in the command file based upon the test result.

The 26 Network/S variables are identified by the letters **A** through **Z**, prefixed with a "%". JCWs and MPE/iX variables are identified by a "!" prefix. For example:

```
#RJIF !MYJCW = 1 THEN label
```

tests a JCW called "MYJCW" and:

```
#RJIF %D = 10 THEN label
```

tests the Network/S variable "D".

The values used in the tests are treated as 16-bit unsigned integers.

The label must begin with an alphabetic character and may consist of no more than 16 alphanumeric characters and must be defined using **#RJLABEL**.

If the evaluated expression is true, execution will continue with the statement immediately following the specified label; otherwise, execution will continue with the next command in the command file.

See Chapter 4, *Other Topics*, for information about Job Control Words used by Network/S.

Syntax:

```
#RJIF {%var|!jcw|value} {<|>|=|>=|<=|<>} {%var|!jcw|value} THEN label
```

RJIN

Transmit a File to a Remote Host

The **#RJIN** command is used to initiate a data transfer from the local computer to the remote host system.

The format of the command is:

```
#RJIN [Data Source] [;option] [;option] ...
```

The parameters to **#RJIN** include:

JCW, Variable and Indirect File substitution will be performed on the text of the command prior to its evaluation. JCWs and MPE/iX variables are identified by a "!" prefix, Variables are identified by a "%" prefix and Indirect Files are identified by a "^" prefix. When an Indirect File is specified, the first record of the file will be read and substituted for the filename in the command. For example, to substitute the variables "M" and "S" in the **WAIT** option, enter:

```
#RJCALC %M = 2
#RJCALC %S = 30
#RJIN filename;WAIT=%M,%S
```

This evaluates to:

```
#RJIN filename;WAIT=2,30
```

Operation Note

Multiple **#RJIN** commands may be executed consecutively by Network/S, resulting in a single data stream being sent to the remote system. Due to this behavior, each **#RJIN** command will read and transmit all but the *last*

block of data read from the input file. The last block is held in abeyance until the next Network/S command is read. If the next command is another **#RJIN**, the first file's final block is sent with an **ETB** terminator (intermediate) and the second file's data follows, unless **ETX=YES** is specified (see **#RJIN ETX** (page 3-41).) Otherwise, the final block is terminated with an **ETX** and the next command is executed.

As a result of this behavior, Network/S does not 'finish' an **#RJIN** until the next command has been read. This will cause the file transfer summary and statistics to be displayed after the next command has been read and displayed. Thus, you may see summary **#RJIN** statistics following the display of an **#RJOUT** command! Note that this behavior may be inhibited by specifying **VERBOSE=NO** and **SHOW=NO** on the **#RJLINE** command.

When invoking Network/S with **$STDINX** as the default command file, be careful to note that an **#RJIN** has not completed when the next "#" prompt has been displayed - Network/S will not finish the transfer until the next command has been entered. For this reason, when end-of-file is encountered on an **#RJIN** file, a warning message is displayed, indicating the amount of data remaining to be sent.

To finish the current transfer, enter an **#RJEOD** command at the prompt or use the **EOD** option with the **#RJIN** command.. See the description for **#RJIN EOD** (page 3-41) or **#RJEOD** (page 3-25) for more information.

Data Source

The **#RJIN** command can specify that data be read from a file or from an SL or XL interface procedure in the following form:

```
#RJIN [filename | * | { @procedurename [ (lib) ] } ]
```

If *filename* is omitted, data will be read from the default input file, designated **RJEIN**, as specified in the command line.

If *filename* is specified, it must be a valid MPE filename. If *filename* refers to the current command file, Network/S will read data records from the point following the current **#RJIN** command up to, but not including, the next line starting with the "#" character, which will be used as the next command. For example, if the file **MYCMDS** contains:

```
#RJLINE 3780; ...
#RJIN MYCMDS
//LOGON IBM style
//MORE IBM jcl
//STILL more IBM jcl
#RJOUT
#RJEND
```

you could enter:

```
:RUN NETWORK.NETWORKS.TELAMON;INFO="MYCMDS"
```

When Network/S opens the **#RJIN** file **MYCMDS**, it will determine that the data file is the same file being used as the current command file. Network/S will correctly read and send, in this example, the three "//" JCL records. The **#RJIN** execution will stop at the next ***#xxx*** command found, the **#RJOUT** command in this case.

Note that this effect can also be obtained as follows:

MYCMDS contains:

```
#RJLINE 3780; ...
#RJIN
//LOGON IBM style
//MORE IBM jcl
//STILL more IBM jcl
#RJOUT
#RJEND
```

Network/S is run:

```
:FILE RJECOM=MYCMDS
:FILE RJEIN=MYCMDS
:RUN NETWORK.NETWORKS.TELAMON;PARM=5
```

or:

```
:RUN NETWORK.NETWORKS.TELAMON;INFO="MYCMDS,MYCMDS"
```

or:

```
:NERJE MYCMDS,MYCMDS
```

When Network/S accesses the default **#RJIN** file, it will note that this file matches the default **RJECOM** file and will behave as above.

If the default input file is **$STDINX**, Network/S will prompt the user at the screen with the "#?" sequence to indicate that data is expected.

If ***** is specified as the input file, data will be read from the current command file.

If no source is specified, data will be read from **$STDINX**, or the file associated with **RJEIN**.

If the designated file is empty, the **#RJIN** command is ignored.

RJIN procedure

The **#RJIN** command normally reads its input from a file identified by the item located between the **#RJIN** command and the first semi-colon (**;**) or the end-of-line.

An optional technique involves the use of a user-written library procedure, located in an SL (Compatibility Mode) or an XL (Native Mode.)

This procedure is called in place of the **FREAD** intrinsic, whenever the **FREAD** intrinsic would be called, once for each logical record in the input data file.

When specifying an external procedure, the syntax of the **#RJIN** command is:

```
#RJIN @procedurename [ ( lib ) ]
```

Network/S will attempt to load the specified procedure, in the exact specified case, from the SL or XL file specified by *lib.*

When running Network/S on an MPE/V based HP 3000 or in Compatibility Mode on an MPE/XL or MPE/iX based HP 3000, *Lib* can be one of the following:

Lib	SL Search sequence	
---	---	
G	SL.*logongroup*.*logonaccount*	then
	SL.PUB.*logonaccount*	then
	SL.PUB.SYS	
P	SL.PUB.*logonaccount*	then
	SL.PUB.SYS	
S	SL.PUB.SYS only	
GX	SL.*appgroup*.*appaccount*	then
	SL.PUB.*appaccount*	then
	SL.PUB.SYS	
PX	SL.PUB.*appaccount*	then
	SL.PUB.SYS	

where *logongroup* and *logonaccount* refer to the user's logon group and account and *appgroup* and *appaccount* refer to the location where the Network/S application resides, by default, NETWORKS and TELAMON respectively. Note that if Network/S has been placed in PUB.SYS, the only meaningful options for *Lib* are G, P and S since GX and PX would both refer to PUB.SYS.

When running the Native Mode version of Network/S on an MPE/iX based HP 3000, Network/S will attempt to load the specified procedure from the XL file identified by *lib*, which must be a valid filename.

See Appendix D, *Procedure Examples*, for an example **#RJIN** procedure.

INCODE

The **INCODE** parameter is used to identify the character set for the data to be sent by this **#RJIN** command. The Engine uses this information the translate the data before it is sent. The translation is performed according to the **#RJLINE**... **LINECODE** setting and is done in such a way that all data sent is of the same character set. The possible values are:

ASCII	The data is ASCII and is converted if **LINECODE=EBCDIC**
EBCDIC	The data is EBCDIC and is converted if **LINECODE=ASCII**
BINARY	The data is not to be converted during transmission (**XPARENT=YES** is recommended)

It is the responsibility of the remote system to perform the appropriate translation upon receipt, if any translation is necessary.

Syntax:

```
#RJIN ... ;INCODE={ASCII | EBCDIC | BINARY}
```

Default:

```
#RJIN ... ;INCODE=ASCII
```

XPARENT

The **XPARENT** parameter is used to inform the Engine that the data to be sent may contain characters that can be misinterpreted as bi-sync control characters. When set to **YES**, the Engine will 'protect' certain characters before sending them by inserting a Data-Link-Escape (**DLE**) character in front of each one found.

If **XPARENT=YES** is selected and the effective **MAXRPB** is more than 1, Network/S will concatenate multiple records together in the transmission block before sending them. If the receiving system requires a **DLE-IUS** separator sequence between logical records, the following sequence will instruct Network/S to do so:

```
XPARENT=YES,IUS=YES
```

Syntax:

```
#RJIN ... ;XPARENT={YES[,IUS={YES | NO}] | NO}
```

Default:

```
#RJIN ... ;XPARENT=NO
```

For transparent transmissions, the defaults are as follows:

```
#RJLINE 2780...
#RJIN ... ;XPARENT=YES,IUS=YES

#RJLINE 3780...
#RJIN ... ;XPARENT=YES,IUS=NO
```

COMPRESS

The **COMPRESS** parameter is used to control compression of multiple blank sequences in the data to be sent. If set to **YES**, any sequence of blanks in excess of two in a row will be compacted to improve transmission efficiency. If set to **NO**, no compression will take place.

Note that the remote system *must* be able to decompress the data for this option to be of any use.

Note also that if **XPARENT=YES** is selected, no compression can or will be performed.

Finally, note that many host systems cannot accept compressed data in the logon step. It is usually necessary to disable compression during the initial **#RJIN** step involved with sending the logon sequence to the remote system. The reason for this is that with some host systems, compression is not performed automatically. It must be requested by some indicator in the logon sequence. Consequently, if the logon sequence is itself compressed, the host may not be able to recognize the logon data.

Compression is an efficiency feature, not a requirement for bi-sync communications - virtually all host systems and 3780 emulators can handle uncompressed data. However, when in doubt, set **COMPRESS=NO**!

Syntax:

```
#RJIN ... ;COMPRESS={YES | NO}
```

Default:

```
#RJLINE 2780; ...
#RJIN ... ;COMPRESS=NO

#RJLINE 3780; ...
#RJIN ... ;COMPRESS=YES
```

TRUNCATE

The **TRUNCATE** parameter is used to control deletion of trailing blanks from the data to be sent. If set to **YES**, trailing blanks will be removed and a special terminator character will be placed after each record to mark the end of the record. If set to **NO**, no truncation will take place.

Note that if **XPARENT=YES** is selected, no truncation can or will be performed.

Note also that with **2780** emulation, an additional character, an End-of-Media (**EM**), is added to any truncated record to indicate that truncation took place. If no truncation was possible, the **EM** is not added.

Syntax:

```
#RJIN ... ;TRUNCATE={YES | NO}
```

Default:

```
#RJIN ... ;TRUNCATE=YES
```

MAXSIZE

The **MAXSIZE** parameter is used to specify the amount of data to be retrieved from the Data Source for each record to be sent. This value defaults to the logical record width of the specified data file.

When **MAXSIZE** has been specified and fewer than the indicated number of bytes are read in any record, Network/S will pad the short record up to **MAXSIZE** bytes, using ASCII or EBCDIC blanks or binary zeroes, according to the **INCODE** parameter.

If it is desirable to override the **MAXRPB** setting in the **#RJLINE** command, it can be done so using the **MAXSIZE** keyword. The new max-records-per-block value will have effect only during the execution of this **#RJIN** command. For example, if **MAXRPB=10** and you'd like to send the current file using one record per block, enter:

```
#RJIN ... ;MAXSIZE=,1
```

Syntax:

```
#RJIN ... ;MAXSIZE=[{-bytes | +words}] [,max records per block]
```

Default:

```
#RJIN file;MAXSIZE={input file's record width}
#RJIN @procedurename;MAXSIZE=-80
```

REC

Normally, the **#RJIN** command will send the entire contents of the specified input disc file. The **REC** option of the **#RJIN** command provides the means to read and transmit a subset of the file. Two parameters may be specified with this option: the starting record number and the ending record number. If **REC** is specified, the following defaults apply:

starting record number: 0 - disc files are numbered starting with zero
ending record number: (EOF-1) - e.g. for a 20 record file: 19

For example, to send the first three records of file DATA, enter:

```
#RJIN DATA;REC=0,2
```

or

```
#RJIN DATA;REC=,2
```

To send all records following record number ten (the 11th record in the file), enter:

```
#RJIN DATA;REC=10
```

To send the fifth (record #4) through eighth (record #7) records, enter:

```
#RJIN DATA;REC=4,7
```

Syntax:

```
#RJIN ... ;REC=[start rec] [,end rec]
```

Default:

```
#RJIN ... ;REC=0,EOF-1
  (e.g. for a 20 record file: REC=0,19)
```

NOETB

Normally, each intermediate data block (that is, each block but the last in the transmission) is terminated with the **ETB** (End of Transmission Block) character, with the last block terminated with an **ETX** (End of Text.) Certain host systems do not support the use of the **ETB** character and require that each data block be terminated with an **ETX**. The **NOETB** keyword, when specified, will cause **ETX**'s to be used on each data block transferred.

Syntax:

```
#RJIN ... ;NOETB={YES | NO}
```

Default:

```
#RJIN ... ;NOETB=NO
```

NOITB

Normally, each logical record in a non-transparent transmission block is delimited with (terminated by) an **ITB** sequence: the **RS** character in 3780 and the **US** character in 2780. For example, in 3780 a data block might appear as:

```
STX data RS data RS data RS data RS ETX
```

Some systems do not expect the **ITB** sequence after the last record in the block. e.g. the above block should appear as:

```
STX data RS data RS data RS data ETX
```

The **NOITB** option will cause Network to skip the *last* **ITB** sequence in each block.

If **NOITB=ALL** is selected, the transmission block will be constructed with no ITB characters at all, similar to the effect of setting **XPARENT=YES**. e.g. the above block should appear as:

STX ***data data data data*** **ETX**

If the **ALL** option is used, it is very likely that **TRUNCATE=NO** and **COMPRESS=NO** should be set also - the receiving system would otherwise have no way of deblocking the data if variable length records are placed in the block.

Note that this option is incompatible with the **REBLOCK** option and is disabled if **REBLOCK** is specified.

Syntax:

```
#RJIN ... ;NOITB={YES | NO | ALL}
```

Default:

```
#RJLINE 3780;...
#RJIN ... ;NOITB=NO

#RJLINE 2780;...
#RJIN ... ;NOITB=YES
```

WAIT

In bi-sync communications, whenever a sender wishes to initiate transmission of a data set, it must first indicate to the remote system its intention to do so. This is referred to as 'bidding' for the line.

The **WAIT** parameter is used to specify how long Network/S will wait for its bid request to be acknowledged. If no answer is received in the specified time interval, or if the request is explicitly denied, Network/S will return an error indicating that the specified file could not be sent.

The **WAIT** timer is also used to time inactivity during a file transfer.

Specifying **WAIT=0** will cause Network/S to wait indefinitely.

Syntax:

```
#RJIN ... ;WAIT=[minutes] [,seconds]
```

Default:

```
#RJIN ... ;WAIT=3,0
```

REBLOCK

Normally, Network/S attempts to put up to **MAXRPB** logical records into a transmission block before sending that block to the remote system. For example, with a maximum buffer size of 512 bytes, six transparent 80-byte records would normally fit. 480 data bytes would be sent with this block with the seventh record appearing as the first record of the second block.

If **REBLOCK** is set to **YES**, Network/S will pack partial records into each transmission block. In the above example, the first six records would take up the first 480 bytes of the 512 byte block and the remaining 32 bytes would be comprised of the first 32 bytes of the seventh record. The second data block would then start with the remaining 48 bytes of the seventh record, followed by the eighth through 12th records in their entirety followed by the first 64 bytes of the 13th record, and so on.

If non-transparent data is sent, the same space savings occur as each logical record is delimited with an ITB character (**RS** or **US**). If **REBLOCK** is **YES** and a block is filled by a partial record, that partial portion is *not* terminated with the ITB. When the remaining portion of the record is finally sent, the ITB is added. In this fashion, the receiving system can re-construct the logical record by accumulating pieces until the ITB has been found.

Do not use this option unilaterally - if the receiving system cannot handle records split across block boundaries, the data file received will not match the file sent.

Syntax:

```
#RJIN ... ;REBLOCK={YES | NO}
```

Default:

```
#RJIN ... ;REBLOCK=NO
```

ROUTE

The **ROUTE** parameter is used to provide a means to send routed data to a remote 3780 emulator.

Normally, it is not necessary to specify routing codes when sending data to a host computer system - the routing codes are used by the actual 3780 terminals and emulators to distinguish between *received* list and punch output. However, if you are communicating with another 3780 emulator which requires routed output (HP's RJE and Network/S do not require routed output), it will be necessary to insert the routing code before the first record of the data set to be sent.

To route data to the remote system's list device, enter:

```
#RJIN ... ;ROUTE=LIST
```

This will cause a **DC1** to be inserted as the routing code.

To route data to the remote system's punch device, enter:

```
#RJIN ... ;ROUTE=PUNCH
```

This will cause a **DC2** to be inserted as the routing code. Since there are two different routing codes for directing output to a punch device, it may be necessary to override the **DC2** code to specify the alternate **DC3** punch routing code. To do so, enter:

```
#RJIN ... ;ROUTE=PUNCH3
```

Syntax:

```
#RJIN ... ;ROUTE={LIST | PUNCH[2|3]}
```

Default:

No routing codes sent.

The **ROUTE** option is available only during 3780 emulation.

EOD

Specifying the **EOD** option will cause Network/S to automatically send an End-Of-Transmission (**EOT**) code at end-of-file. This has the same effect as issuing an **#RJEOD** command immediately after the **#RJIN**.

ETX

Specifying the **ETX** option will cause Network/S to automatically append an End-of-Text (**ETX**) code to the final transmission block at end-of-file, even if the next command is another **#RJIN**. Normally, an End-of-Transmission-Block (**ETB**) code is appended if the next command to be executed is another **#RJIN**.

Use this option if the receiving system requires each file sent to be delimited with an **ETX** code.

Syntax:

```
#RJIN ... ;ETX={YES | NO}
```

Default:

```
#RJIN ... ;ETX=NO
```

ECHO

When set to **YES**, the **ECHO** option will cause each record read from **$STDINX** in batch to be displayed upon **$STDLIST**, as it is read.

Data read from **$STDINX** in Interactive mode will not be explicitly echoed, as it is automatically echoed as it is input.

Syntax:

```
#RJIN ... ;ECHO={YES | NO}
```

Default:

```
#RJIN ... ;ECHO=NO
```

CCTL

The **CCTL** option on the **#RJIN** command will cause Network/S to convert embedded carriage control directives to their IBM equivalent before transmission. In this fashion, Network/S can send print files to remote 2780/3780 emulators.

This feature assumes that the first byte of each record read is a carriage control character - no test is performed to verify that the file has the CCTL attribute.

To send routed print files, use

```
#RJIN file;ROUTE=LIST;CCTL=YES
```

To send mixed routed print and punch files, use:

```
#RJIN list1;ROUTE=LIST;CCTL=YES;ETX
#RJIN punch;ROUTE=PUNCH;ETX
#RJIN list2;ROUTE=LIST;CCTL
#RJEOD
```

The three files will be sent in a single transmission, with **ETX**-terminated blocks separating the individual data sets.

Syntax:

```
#RJIN ... ;CCTL={YES | NO}
```

Default:

```
#RJIN ... ;CCTL=NO
```

Example

To send an 80-byte ASCII text file, named **DATA**, to a remote host in non-transparent mode, using space-compression, enter:

```
#RJIN DATA;XPARENT=NO;COMPRESS=YES;MAXSIZE=-80
```

Note that if the 3780 protocol is being used, the **COMPRESS** keyword need not be specified as compression is performed by default in non-transparent mode. Note also that if the input data file has a record width of 80 bytes, the **MAXSIZE** keyword need not be specified, as Network/S uses the disc record width as its default. Finally, note that **XPARENT** is, by default, disabled. In 3780, then, the above command could be entered:

```
#RJIN DATA
```

and in 2780, it would be:

```
#RJIN DATA;COMPRESS=YES
```

since, in 2780, compression is by default not enabled.

To send an HP 3000 program file, named PROGRAM, to another HP 3000 running either HP's RJE or Network/S, enter:

```
#RJIN PROGRAM;INCODE=BINARY;XPARENT=YES;MAXSIZE=128
```

Again, note that the **MAXSIZE** parameter may be omitted since Network/S uses the file's record width as its default value. Note that with **INCODE=BINARY**, no compression or truncation will be performed regardless of the implicit or explicit settings of the other keywords.

RJINFO

Display Line Information

The **#RJINFO** command generates a display showing the configuration for the current communications setup.

If no Engine has been accessed, the display looks like:

```
#RJINFO
************************************************
*-L-I-N-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y*
************************************************
*  PROGRAM: NETWORK.NETWORKS.TELAMON            *
*  VERSION: A.03.00                             *
************************************************
*  RJCONTINUE ABORT                             *
************************************************
```

If an Engine *has* been accessed, the display looks like:

```
   #RJINFO
   ************************************************
   *-L-I-N-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y*
   ************************************************
 1 *  PROGRAM: NETWORK.NETWORKS.TELAMON            *
 2 *  VERSION: A.03.00                             *
 3 *  ENGINE:  A.7.1                               *
 4 *  DEVICE:  106                  SPEED: 1920    *
 5 *  RIN:     1,NETWORKS                          *
 6 *  EMULATE: 3780         LINECODE: EBCDIC,CRC   *
 7 *  TABLE:   default                             *
 8 *  SBIN:    NSF71BIN.NETWORKS.TELAMON           *
   *  ABIN:    NSV71BIN.NETWORKS.TELAMON           *
 9 *  DIALER:  V.25,ASCII                          *
10 *  PHONE:   555-1234                            *
11 *  ONLINE                STATE: CONTROL         *
12 *  TRANSMIT INFO: MAX:  6116 CUR:     0         *
13 *   RECEIVE INFO: MAX:  8960 CUR:     0         *
14 *  LIMITS:                                      *
   *     SYNs:     4        ENQs:     6           *
   *     NAKs:     6        TTDs:     1*3 sec     *
15 *  TIMERS:                                      *
   *     DIAL:    20*4 sec  CRASH:  600 sec       *
   *     RTSCTS:   0 ms     PACKET:   3 sec       *
16 *  TRACE INFO: Auto                             *
   *  ASYNC I/O:                                   *
17 *     TIMEOUTS:       0  OTHER:        0       *
   ************************************************
18 *  RJCONTINUE ABORT                            *
   ************************************************
```

The lines in this display have the following definitions:

1. The fully qualified name of the current application
2. The current software version
3. The current Network Engine firmware version
4. The device name|number and asynchronous speed (**#RJLINE** & **RJECLINE DEV**=)

5. The RIN number and password, if specified (**#**RJLINE & RJECLINE RIN=)
6. The current emulation and linecode values (**#RJLINE** & **RJECLINE LINECODE**=)
7. ASCII-to-EBCDIC translation data file (**#RJLINE** & **RJECLINE TABLE**=)
8. For downloadable Engines, the names of the Asynchronous and Synchronous firmware files (**RJECLINE SBIN**= and **ABIN**=)
9. For dial-out access, the modem dialer type (**#RJLINE** & **RJECLINE MODEM**=)
10. For dial-out access, the phone number (**#RJLINE CONNECT**=**DIAL**)
11. Current Engine state
12. Transmit buffer info, maximum and current
13. Receive buffer info, maximum and current
14. Various configurable counters and limits
    - Configured SYN character count (**RJECLINE SYNS**=)
    - Configured ENQ retry counter (**RJECLINE ENQS**=)
    - Configured NAK retry count (**RJECLINE NAKS**=)
    - Configured TTD pause interval (**RJECLINE TTDS**=)
15. Various configurable timers
    - Maximum time to wait for auto-dialer response (**RJECLINE DIALWAIT**=)
    - Computer inactivity timer (**RJECLINE CRASH**=)
    - Engine-enforced RTS-CTS turnaround timer (**RJECLINE RTSCTS**=)
    - Computer-to-Engine data packet timer (**RJECLINE DATATIMER**=)
16. Current tracing option (**#RJLINE** & **RJECLINE TRACE**=)
17. Various computer-to-Engine counters
    - Recoverable software timeouts
    - Other recoverable I/O errors
18. **#RJCONTINUE** option in effect

Syntax:

```
#RJINFO
```

RJIO

Interact with Remote Host

The **#RJIO** command provides a quick method for sending a small amount of text to the remote host in anticipation of the receipt of output for **$STDLIST**. The form

```
#RJIO text
```

is functionally equivalent to:

```
#RJIN *;COMPRESS=NO;TRUNCATE=NO;MAXSIZE=-80
text
#RJEOD
#RJOUT $STDLIST;WAIT=3,0
```

#RJIO is used generally to log on to a remote host system and to issue remote operating system commands.

An alternate method for **#RJIO** is to input any line of text at the "#" prompt that begins with a special (non-alpha/numeric) character. For instance:

```
#RJIO /*SIGNON USERID
```

can be entered as:

```
/*SIGNON USERID
```

at the "#" prompt.

Syntax:

```
#RJIO text
```

or, if *text* starts with a non-alphanumeric character:

```
text
```

RJLABEL

Identify a Branch Label

The **#RJLABEL** command is used to identify locations in the command file that can be used as the target of a branch command (**#RJIF**, **#RJGOTO** and **#RJCONTINUE**).

The label must begin with an alphabetic character and may consist of no more than 16 alpha-numeric characters.

Syntax:

#**RJLABEL** ***label***

RJLINE

Define Data Communications Link

The **#RJLINE** command is used to specify the communications parameters to be used for the next link to a remote processor. These parameters, for the most part, describe features that will remain in effect for the duration of the communications link. The options are:

JCW, Variable and Indirect File substitution will be performed on the text of the command prior to its evaluation. JCWs and MPE/iX variables are identified by a "!" prefix, Variables are identified by a "%" prefix and Indirect Files are identified by a "^" prefix. When an Indirect File is specified, the first record of the file will be read and substituted for the filename in the command. For example, to substitute the variables "M" and "S" in the **WAIT** option, enter:

```
#RJCALC %M = 2
#RJCALC %S = 30
#RJLINE 3780;WAIT=%M,%S
```

This evaluates to:

```
#RJLINE 3780;WAIT=2,30
```

To load the contents of the file **PHONENUM**, enter:

```
#RJLINE 3780;CONNECT=DIAL,"^PHONENUM"
```

If the first record of **PHONENUM** contains 555-1234, this evaluates to:

```
#RJLINE 3780;CONNECT=DIAL,"555-1234"
```

Note that trailing blanks will be removed from the record read from the Indirect File.

2780 | 3780

The Synchronous Network Engine is capable of emulating either a **2780** or a **3780** Remote Job Entry station. This value will be usually determined by requirements of the remote host computer system. If the remote host will be using the **2780** protocol, **2780** must be specified in the **#RJLINE** command.

If a choice exists, **3780** is preferred as it is the more efficient of the two emulators.

Syntax:

```
#RJLINE {2780 | 3780}
```

Default:

```
#RJLINE 3780
```

LINECODE

The **LINECODE** parameter is used to specify the line transmission code to be used. The possible values are **ASCII** and **EBCDIC** and *must* match the codes used by the remote host.

When **EBCDIC** is selected, a Cyclic Redundancy Check (CRC) checksum scheme is employed during data transfer. When **ASCII** is selected, a Longitudinal Redundancy Check (LRC) scheme is used for non-transparent transmissions; for transparent **ASCII ,CRC** is used automatically. To override these defaults, append either **CRC** or **LRC**, as is appropriate, to the transmission code.

Syntax:

```
#RJLINE ... ;LINECODE={EBCDIC|ASCII}[,{CRC|LRC}]
```

Default:

```
#RJLINE ... ;LINECODE=ASCII,LRC
```

CONNECT

The **CONNECT** keyword, if specified, indicates that a switched-line, dialup connection is to be attempted, either originating from the local computer or from the remote system.

If the **CONNECT** keyword is omitted, a direct-connect link is assumed.

Originating a Call

To specify a connection originating from the local computer, enter:

CONNECT=DIAL,"*phone number*"[,*dialer type*]

The *phone number* entry should be the exact string of text to be used by the modem, or by the operator should this be a manual dial.

The *dialer type*, if specified, should be one of the following values:

DTR[*] Manual dial (default). The Engine will raise the Data-Terminal-Ready signal to the modem and will then wait for a connection to occur. With this setting, a message will appear on $STDLIST and on the system console requesting that the specified phone number be dialed. The console message will require a **:REPLY** for Network/S to proceed. Append ***** to the **DTR** keyword to suppress the console request.

UDS[*] Auto-dial using a Universal Data Systems modem, either a `UDS201C/D` or a `UDS208B/D`. These modems can be automatically dialed from the Engine. The specified phone number will be passed to the modem after editing. Valid characters include the numeric digits and "`*#TP,:`". The UDS modems require a dial command in the form:

`D`*`phone info`*

Note that the Engine will supply the "`D`" - the remainder of the information will be taken from the specified phone number.

Use a "`T`" prefix to force Tone dialing, a "`P`" prefix to force Pulse dialing, a "`:`" to pause for 5 seconds and a "`,`" to cause the modem to wait for a second dial tone.

Space (" ") and dash ("`-`") characters may be included to improve readability - they will be removed before the number is sent to the modem.

Append ***** to the **UDS** keyword to force tone dialing on all numbers.

SADL Auto-dial using a Racal-Vadic modem that supports the Synchronous Auto Dial Language dialing method. These modems can be automatically dialed from the Engine. The specified phone number will be passed to the modem after editing. Valid characters include the numeric digits and "*#K".

If necessary, use modem Option 5 to force Tone or Pulse dialing. A "K" will either pause for 5 seconds or wait for a second dial tone, depending upon modem Option 6.

Space (" ") and dash ("-") characters may be included to improve readability - they will be removed before the number is sent to the modem.

V.25[**,ASCII**] Auto-dial using any V.25bis compatible synchronous auto dial modem.

By default, the Network Engine uses the EBCDIC character set while issuing V.25 dialer instructions. If your modem requires ASCII, append **,ASCII** to the modem keyword.

The specified phone number will be passed to the modem after editing. Valid characters include the numeric digits and "*#W:,KPT&!.ABCD". See your modem's manual for a description of these options.

HAYES For "F" series Synchronous Network Engines, those with the first letter of the Engine Version response reported as "F". Auto-dial using Hayes-style AT commands.

The "F" series Engines are capable of switching from asynchronous mode for dialing to synchronous mode for online mode. The Engine will initialize the modem by sending ATH0E1V1Q0. Upon receiving the OK response, dialing will commence. The Engine will supply the "AT" prefix - any and all remaining configuration and dialing instructions will have to be supplied with the phone number. e.g.

```
CONNECT=DIAL,"&F&M1&C0&R1&S1DT5551234",HAYES
```

This example should work on most Hayes-compatible modems.

801 Auto-dial using the Bell 801 protocol. This option requires a special version of the Synchronous Network Engine and a custom RS-366 interface to the auto-dial unit.

The specified phone number will be passed to the modem after editing. Valid characters include the numeric digits and "*#E/DU". "E" is the End-of-Number (EON) code, "/" is the Separator character, "D" is the Delay character and "U" is the User code. See your dialer's manual for a description of these options.

'Phone number' *must* be specified if the modem type is not **DTR**.

Syntax:

```
#RJLINE ... ; &
#  CONNECT=DIAL,"phone number"[,{DTR[*]|SADL|UDS[*]|V.25[,ASCII]|HAYES|801}]
```

Example:

```
#RJLINE ... ;CONNECT=DIAL,"5551234",SADL
```

or:

```
#RJLINE ... ;CONNECT=DIAL,"5551234",V.25,ASCII
```

See Appendix C, *Modem Configurations*, for more information on specific modem types.

Answering a Call

To specify a connection originating from the remote system, enter:

```
CONNECT=ANSWER
```

The Engine will wait for either an incoming call to arrive or until the time limit, specified by the **WAIT** parameter, expires.

The Engine will normally raise Data-Terminal-Ready (**DTR**) to the modem while awaiting the incoming call. If your modem is configured to automatically dial on **DTR** going high, this will cause problems. Appending an asterisk (*****) to the **ANSWER** keyword will cause the Engine to first wait for Ring-Indicator (**RI**) to come on before raising **DTR** to the modem. This way, **DTR** isn't raised until the phone rings, at which point the auto-dial feature of the modem *should* be disabled.

Syntax:

```
#RJLINE ... ;CONNECT=ANSWER[*]
```

SELECT

The **SELECT** keyword can only be used with a Dual Modem Network Engine. It provides the means to identify the modem to be used for the current session.

The Dual Modem Network Engine provides two modem connections into a single computer terminal port. These modems are identified by the port identifier on the Engine, **MODEM A** or **MODEM B**.

For auto-dialing, the **SELECT** keyword is used to choose one of the two modem ports. To select the modem attached at **MODEM A**, enter:

```
#RJLINE ... ;SELECT=MODEMA
```

To select the modem attached at **MODEM B**, enter:

```
#RJLINE ... ;SELECT=MODEMB
```

For answering incoming calls, a third option, **EITHER**, is available to instruct the Engine to answer an incoming call on either modem.

If the two modems use different dialing protocols, you will have to identify the modem type along with the port when out-dialing. For example, if you have a UDS modem attached to **MODEM A** and a modem which uses V.25 dialing on the **MODEM B** port, you would enter:

```
#RJLINE ... ;CONNECT=DIAL,"5551234",UDS;SELECT=MODEMA
```

and

```
#RJLINE ... ;CONNECT=DIAL,"5551234",V.25;SELECT=MODEMB
```

Alternately, you can add the following lines to the **RJECLINE** configuration file:

```
MODEMA=UDS
MODEMB=V.25
```

Then, when **MODEMA** is **SELECT**ed, the **UDS** modem will be supplied as the default and the **V.25** modem will be used when **MODEMB** is **SELECT**ed.

Syntax:

```
#RJLINE ... ;SELECT={MODEMA | MODEMB | EITHER}
```

Default:

```
#RJLINE ... ;SELECT=MODEMA
```

MAXRPB

The **MAXRPB** keyword specifies the maximum records per block that can be sent during execution of subsequent **#RJIN** commands. The default values are:

TABLE 3-1.

	2780	**3780**
Transparent	4	6
Non-Transparent	7	255

These defaults are partially based upon the default maximum buffer size: 400 bytes for 2780 and 512 bytes for 3780. Four transparent 80-byte records (plus some overhead) can fit in a 400-byte 2780 buffer and six transparent 80-byte records can fit in a 512-byte 3780 buffer. For non-transparent data (data that can be either compressed or blank-truncated), more records may fit into the buffer.

The Synchronous Network Engine supports buffer sizes of up to 4,096 bytes for either protocol. The default buffer size matches the defaults for HP's RJE. To specify a larger buffer size, an additional parameter value can be specified in the **MAXRPB** setting, as follows:

MAXRPB=[*block factor*] [,*buffer size*]

For example, to specify a **MAXRPB** value of 20 with a maximum buffer size of 1,024 bytes, you would enter:

MAXRPB=20,1024

To use the default **MAXRPB** setting while increasing the buffer size to 2,048 bytes, you would enter:

MAXRPB=,2048

Note that the **MAXRPB** blocking factor value applies to all subsequent **#RJIN** commands, although it may be overridden on an individual basis in each **#RJIN** command using the **MAXSIZE** option (page 3-37).

Syntax:

#**RJLINE ... ;MAXRPB=[*block factor*] [,*buffer size*]**

DEV

The **DEV** keyword is used to identify the logical device to which the Synchronous Network Engine is attached and to specify the speed to be used between the Engine and the local HP 3000's terminal port. The format is:

DEV=[*logical device name/number*] [,*speed*]

If the *logical device name* is omitted, the Network/S will attempt to open a port with the class-name **RJLINE** and formal designator **RJELINE**. If a **:FILE** equation is issued for **RJELINE**, that will override the default. If the *speed* parameter is omitted, a value of 960 cps will be used.

Note that the speed setting *must* match the DIP-switch settings on the Engine, positions 1-4. If these values do not match, Network/S won't be able to communicate with the Engine!

Syntax:

#**RJLINE ... ;DEV=[*logical device name/number*] [,*speed*]**

Default:

#**RJLINE ... ;DEV=RJLINE,960**

Note that the *logical device name* and *speed* may be specified in the **RJECLINE** file.

XEND

During normal communications, receipt of a **DLE-EOT** immediately precedes a line disconnect. This usually indicates that the remote host is finished, either normally or abnormally, and is preparing to cease communications.

There are certain circumstances, dictated by the remote system, where **DLE-EOT** is *not* to be interpreted this way. The **XEND** keyword will instruct Network/S to treat a **DLE-EOT** sequence like an **EOT** sequence, which is usually used to separate multiple transmissions during the communications session.

Unfortunately, the use of the **XEND** keyword can introduce a problem under certain circumstances. Should the remote system actually attempt to break the connection due to some kind of error, it will probably send a **DLE-EOT** sequence before doing so. In this context, **DLE-EOT** should be interpreted as an error condition. It would be desirable to be able to distinguish between this error condition and the **DLE-EOT** sent in lieu of an **EOT** at the termination of an **#RJOUT** command. Appending an asterisk ("*****") to the **XEND** keyword will cause Network/S to treat **DLE-EOT** sequences which immediately follow a received **ETX** (at the end of an **#RJOUT**) as an **EOT**. **DLE-EOT**'s that occur in any other context will be treated as errors.

Syntax:

```
#RJLINE ... ;XEND[*]
```

Default:

DLE-EOT causes Network/S to shut down the link and terminate.

WAIT

If either **CONNECT=ANSWER, CONNECT=DIAL,"...",DTR** or a direct-connect link is being attempted, the **WAIT** parameter will specify how long Network/S will wait for the connection to be established. The value is specified as follows:

```
WAIT=[minutes] [,seconds]
```

For example, to wait a maximum of 5 minutes for an incoming call, you would enter:

```
#RJLINE ... ;CONNECT=ANSWER;WAIT=5,0
```

or:

```
#RJLINE ... ;CONNECT=ANSWER;WAIT=0,300
```

Note that if any auto-dial option is specified, the modem will dictate the wait period. Most modems wait about 60 seconds for the answering modem to connect before aborting the attempt. The **RJECLINE** parameter, **DIAL-WAIT**, can be used to provide the Network Engine with a maximum wait time for the modem's response.

Specifying **WAIT=0** will cause Network/S to wait indefinitely.

Syntax:

```
#RJLINE ... ;WAIT=[minutes] [,seconds]
```

Default:

```
#RJLINE ... ;WAIT=0,0
```

RIN

A Resource Identification Number (RIN) can be used by Network/S to help prevent conflicts when two or more processes attempt to access the same Engine port at the same time.

Without the use of RINs, the operating system usually prevents any problems from occurring by granting access to the first requesting process; subsequent processes will encounter an unavailable device. However, if Network/S is being run multiple times from within the same job/session via process handling, problems can arise since multiple processes in the same job/session can successfully access the same port.

RINs are lockable resources that can be used to prevent such problems from occurring and the **RIN** keyword is used to attempt to lock such a resource. The MPE command **:GETRIN** must have been previously issued to create a permanent RIN. See the *MPE Commands Reference Manual* for information on this command.

For example, if RIN #32 exists with a password of NETWORKS, you would specify:

```
#RJLINE ... ;RIN=32,NETWORKS
```

If the RIN is available, it will be locked to prevent any other processes from accessing the Engine. The RIN will be released when the Engine port is closed by Network/S.

If the RIN is already locked by another process, Network/S will suspend until the RIN becomes available. If **#RJLINE ... WARN=YES**, messages will be displayed prior to the suspension and after the lock is granted.

Note that a direct relationship between RINs and Engine ports must be strictly maintained. It is only by convention that the RIN locking procedure can successfully prevent problems. One possible scheme to employ would be to include the Engine's device number in the RIN password when acquiring it via the **:GETRIN** command, as follows:

```
:GETRIN LDEV33
RIN: 43
:GETRIN LDEV34
RIN: 47
:
```

We would then use **RIN=43,LDEV33** for DEV=33 and **RIN=47,LDEV34** for DEV=34. Note that the actual RIN numbers are assigned randomly by MPE.

Note also that RINs are permanently assigned. They may be released only via the **:FREERIN** command.

Finally, note that the Global RIN table is defined during system configuration. Your system must have space for any additional RINs before the **:GETRIN** command may be successfully executed.

Syntax:

#**RJLINE ... ;RIN=*rin,password***

Default:

None - no RIN will be locked

PRI

Network/S is normally run using the default priority provided by the job or session - e.g. CS for a session and DS for a job.

Network/S has two options for overriding this setting. If Network/S has been prepared with CAP=PM (or modified using the **MAXCAP** contributed utility), the **PRI** options are:

PRI={NORMAL | HIGH[,*abspri*]}

NORMAL will result in the default job/session priority being used. If **HIGH** is selected, Network/S will by default request absolute priority level 150. If *abspri* is specified, that value will be used instead of the default.

Note: Set absolute priority with care - too low a value will have Network/S directly competing with MPE for resources and can result in a deadlocked system!

If Network/S is running in non-privileged mode, **HIGH** priority will be interpreted as CS priority.

Syntax:

#**RJLINE ... ;PRI={NORMAL | HIGH[,*abspri*]}**

Default:

```
#RJLINE ... ;PRI=NORMAL     {Note that HP's RJE defaults to HIGH}
```

ID

The **ID** keyword specifies the local terminal identification string used by the Engine when a communications link is first made. Use this feature only if required by the remote system.

The value is specified in ASCII and will be translated, if necessary, to match the configured **LINECODE**. If a non-printing character needs to be specified, Hex notation can be used. For example, if the ID needs to be a two byte sequence consisting of the letter “A” followed by a hex 'FA', use

```
#RJLINE ... ;ID="A$FA"
```

If the **LINECODE** is EBCDIC, this will be first translated to:

```
$C1 $FA
```

To represent the “$” character itself, use **$23**.

Syntax:

```
#RJLINE ... ;ID="id"
```

Default:

none

REMID

The **REMID** keyword specifies the remote terminal identification string to be recognized by the Engine when a communications link is first made. Specify this string only if you expect the remote system to send its local ID string when bidding for the line.

The value is specified in ASCII and will be translated, if necessary, to match the configured **LINECODE**. If a non-printing character needs to be specified, Hex notation can be used. For example, if the REMID needs to be a two byte sequence consisting of the letter “A” followed by a hex 'FA', use

```
#RJLINE ... ;REMID="A$FA"
```

If the **LINECODE** is EBCDIC, this will be first translated to:

```
$C1 $FA
```

To represent the “$” character itself, use **$23**.

Syntax:

```
#RJLINE ... ;REMID="remid"
```

Default:

none

MSGFILE

The **MSGFILE** keyword is used to specify an MPE message file to be used for programmatic access.

If the specified file is not fully qualified, Network/S will attempt to look for it in the **MSG** group of the **RJE** account. Otherwise, the specified fully qualified name will be used.

See Appendix F, *Programmatic Access*, for more information on its use.

Syntax:

```
#RJLINE ... ;MSGFILE=MYFILE          Opens MYFILE.MSG.RJE
#RJLINE ... ;MSGFILE=MYFILE.PUB      Opens MYFILE.PUB.logonaccount
```

Default:

no message file

SHOW

The **SHOW** keyword controls the automatic display of configuration and run-time parameters during the execution of Network/S. If set to **YES**, Network/S will print a summary of the command parameters in effect after each **#RJLINE**, **#RJIN** and **#RJOUT** command. If set to **NO**, no such display will be generated.

Syntax:

```
#RJLINE ... ;SHOW={YES | NO}
```

Default:

```
#RJLINE ... ;SHOW=YES
```

VERBOSE

The **VERBOSE** keyword controls the automatic generation of transfer progress and summary statistic displays during the execution of Network/S. If set to **YES**, Network/S will print progress updates during file transfers and print a performance summary after each file transfer command. If set to **SUMMARY**, only the performance summaries will be displayed. If set to **NO**, no such displays will be generated.

Syntax:

```
#RJLINE ... ;VERBOSE={YES | SUMMARY | NO}
```

Default:

```
#RJLINE ... ;VERBOSE=YES      {Commands read from $STDINX}
#RJLINE ... ;VERBOSE=SUMMARY  {Commands read from a file or in batch}
```

WARN

The **WARN** keyword controls the automatic generation of warning messages during the execution of Network/S. If set to **YES**, Network/S will display various informative warning messages during its execution. If set to **NO**, no such displays will be generated.

Syntax:

```
#RJLINE ... ;WARN={YES | NO}
```

Default:

```
#RJLINE ... ;WARN=YES
```

TRACE

The **TRACE** keyword enables the creation of a diagnostic dump file, **XFERDUMP**, for later analysis via the **FMTDUMP** program, accessed using the **#FMTDUMP** command.

If **TRACE** is set to **YES**, Network/S will create the dump file and log all activity between Network/S and the Synchronous Network Engine in use. This information will include all Intrinsic calls in addition to all data transfers. If the dump file already exists, it will be purged and re-built. If **TRACE** is set to **CIR**, **XFERDUMP** will be created as a Circular file, conserving space.

To examine the contents of the dump file, you must use the **FMTDUMP** program, accessed via the **#FMTDUMP** command. **#FMTDUMP** will attempt to open a dump file in the logon group and account. If the dump file is in another location, or has been renamed, the filename can be specified in the ***dumpinfo*** parameter (below).

For example, to examine a local dump file, enter:

```
#FMTDUMP
```

Parameters to **#FMTDUMP** include:

J Display formatted listing on **$STDLIST**, 23 lines a time.

L Direct output to the line printer

S	When specified, **#FMTDUMP** will prompt the user for a search string and a count, *n*, and will start generating output when the *n*'th occurrence of the string is found.
T	Each entry in the formatted output has a time-stamp marking it. If this option is specified, **#FMTDUMP** will prompt for a target time-stamp to search for before beginning the report. The time-stamp value may be specified as an integral number of milliseconds or as seconds.fraction. The presence of a "." in the response indicates the form chosen. If the listing is directed to the line printer, **#FMTDUMP** will prompt for an ending time-stamp as well.
U	Generate a trace listing in User Format, showing the transfer dialog in a format more closely resembling the actual bi-sync transmission.
"*dumpinfo*"	Specify a file other than **XFERDUMP** or a location other than the logon group/account.

The parameters may be combined, separated by commas (,).

Note: Unless displayed in User Format, the trace listing will be difficult to interpret. When necessary, detailed trace analysis can be performed by Telamon, using files created on your system.

To examine a dump file that has been renamed to **OLDDUMP**, enter:

```
#FMTDUMP,"OLDDUMP"
```

To examine a dump file that is located in **PUB.SYS**, additionally specifying the Jump option, enter:

```
#FMTDUMP,J,".PUB.SYS"
```

The output from **#FMTDUMP** will be directed to **$STDLIST** unless the **L** option has been specified. e.g.

```
#FMTDUMP,L
```

will cause the formatted output to be sent to device class **LP** via formal designator **XFERLIST**. If you wish to redirect the output to another device, enter:

```
:FILE XFERLIST;DEV=...
```

before executing **#FMTDUMP**.

The **TRACE** capability is left enabled until Network/S terminates, or until another **#RJLINE** command is entered with **TRACE=NO** specified.

In the absence of an explicit **TRACE** setting, Network/S will open a temporary, circular trace file, named **XFER0DMP**, which will be saved only in the event of an abnormal program termination. When, and if, this file is saved, an appropriate message will be displayed. Setting **TRACE=NO** suppresses the creation of either type of trace file.

This file can then be analyzed using:

```
#FMTDUMP,"XFER0DMP"
```

or

```
#FMTDUMP,0
```

Syntax:

```
#RJLINE ... ;TRACE={YES | NO | CIR}
```

Default:

```
#RJLINE ... ;TRACE=NO
```

RETRY

In the event that there are any asynchronous data-communications errors between the Synchronous Network Engine and the local computer, Network/S will normally attempt up to four times to correct the situation. The **RETRY** keyword provides the means to change this limit.

These errors are most often due to data-overruns on the computer's terminal port. At 9600 baud, under certain circumstances, an ADCC port can be overrun when reading data from the Engine. The overrun error causes Network/S to instruct the Engine to re-transmit the record and will try up to **RETRY** times consecutively before terminating with an unrecoverable error.

Note: This parameter does *not* affect the Network Engine's synchronous behavior. It only applies to the asynchronous link between the Engine and the local computer. The **RJECLINE** parameters **ENQ** and **NAK** control the synchronous retry options.

Syntax:

```
#RJLINE ... ;RETRY=count
```

Default:

```
#RJLINE ... ;RETRY=4
```

REDIAL

The **REDIAL** option, when set to a non-zero value, will instruct Network/S to re-execute the dialing sequence in the event that the previous attempt has failed. If the *wait* parameter is specified, Network/S will pause for *wait* minutes before the next attempt.

If *count* additional attempts fail, the normal error behavior will occur.

Syntax:

```
#RJLINE ... ;REDIAL=count[,wait]
```

Default:

```
#RJLINE ... ;REDIAL=0,1
```

HT

Received data streams are examined for the presence of horizontal tab characters (**HT**, $09-ASCII, $05-EBCDIC.) If a tab-definition record has been previously received, the **HT** characters will be replaced with a suitable number of blanks to skip to the next defined tab position. If no tab-definition record has been received, the **HT** characters will be replaced with spaces. (Tab-definition records appear at the beginning of a received file and start with an **ESC**-**HT**.)

Under certain circumstances, it would be desirable to disable this automatic transformation. Setting **HT=NO** will suppress the conversion of horizontal tab characters.

Syntax:

```
#RJLINE ... ;HT={YES | NO}
```

Default:

```
#RJLINE ... ;HT=YES
```

FF

Received 3780 data streams are examined for the presence of the form-feed character (**FF**, $0C-ASCII/EBCDIC.) These are treated as a record delimiters causing any preceding data to be written to disc as a logical record with suitable carriage control.

Under certain circumstances, it would be desirable to disable this feature. Setting **FF=NO** will suppress this interpretation of form-feed characters.

Syntax:

```
#RJLINE ... ;FF={YES | NO}
```

Default:

```
#RJLINE ... ;FF=YES
```

LF

Received 3780 data streams are examined for the presence of the line-feed character (**LF**, $0A-ASCII, $25-EBCDIC.) These are treated as a record delimiters causing any preceding data to be written to disc as a logical record with suitable carriage control.

Under certain circumstances, it would be desirable to disable this feature. Setting **LF=NO** will suppress this interpretation of line-feed characters.

Syntax:

```
#RJLINE ... ;LF={YES | NO}
```

Default:

```
#RJLINE ... ;LF=YES
```

NL

Received 3780 data streams are examined for the presence of the new-line character (**NL**, $85-ASCII, $15-EBCDIC.) These are treated as a record delimiters causing any preceding data to be written to disc as a logical record with suitable carriage control.

Under certain circumstances, it would be desirable to disable this feature. Many EDI systems, for example, use the **NL** character as an integral part of the data record. Splitting the record at this point renders the data unusable. Setting **NL=NO** will suppress this interpretation of new-line characters.

Syntax:

```
#RJLINE ... ;NL={YES | NO}
```

Default:

```
#RJLINE ... ;NL=YES
```

VT

Received 3780 data streams are examined for the presence of the vertical-tab character (**VT**, $0B-ASCII/EBCDIC.) These are treated as a record delimiters causing any preceding data to be written to disc as a logical record with suitable carriage control.

Under certain circumstances, it would be desirable to disable this feature. Setting **VT=NO** will suppress this interpretation of vertical-tab characters.

Syntax:

```
#RJLINE ... ;VT={YES | NO}
```

Default:

```
#RJLINE ... ;VT=YES
```

CCODE

The definition for the use of User Procedures, as specified in the HP's RJE manual, indicates that Condition Codes are to be used only for signaling End-of-File during the execution of an **#RJIN** input procedure. If any error occurs during the execution of this or any other procedure, the only recourse is to abort, possibly leaving the Network Engine in an on-line state.

Setting **CCODE=YES** will cause Network/S to test the returned condition codes after **#RJIN, #RJOUT** and **#RJSTAT** procedure calls. For **#RJIN** procedures, a CCL code will be treated as an error and normal error processing will be used. For **#RJOUT** and **#RJSTAT** procedures, both CCL and CCG will be treated as errors.

Syntax:

```
#RJLINE ... ;CCODE={YES | NO}
```

Default:

```
#RJLINE ... ;CCODE=NO
```

NOW

By default, execution of the **#RJLINE** command only verifies the specified parameters and accesses the designated **#RJLINE** device. The actual connection to the remote system is initiated only after Network/S encounters the first input/output request, typically an **#RJIN** or **#RJOUT**. Consequently, all messages pertaining to the connection attempt are displayed *after* the parameters of that subsequent **#RJIN/#RJOUT**.

The **NOW** keyword will force the connection to be attempted during the execution of the **#RJLINE** command, as opposed to being deferred until after the first **#RJIN/#RJOUT**. The major benefit of this option is to show the results of the connection attempt in a more natural sequence, immediately following the display of the **#RJLINE** parameters, instead of following the **#RJIN/#RJOUT** parameters. Additionally, connection error checking using **#RJIF** can then be performed immediately after the **#RJLINE** command, where it normally would have to be checked after the first **#RJIN/#RJOUT**.

Syntax:

```
#RJLINE ... ;NOW={YES | NO}
```

Default:

```
#RJLINE ... ;NOW=NO
```

TABLE

The Network Engine performs any and all necessary ASCII-to-EBCDIC and EBCDIC-to-ASCII translations during file transfers. The default translation table used by the Engine was derived from that used by Hewlett-Packard's RJE product. This is also the table supported by the MPE **CTRANSLATE** intrinsic.

The *IBM 3780 Component Description* manual specifies a slightly different set of translations. The majority of the printable characters are translated the same as with HP's table, but there are a few characters that have different values. In particular, the two tables specify different translations for the ASCII exclamation point character (“!”). HP's translation converts this character to an EBCDIC value of 79 decimal, or 4F hex. IBM's translation converts to 90 decimal, or 5A hex. There are other differences, of course, but this character is the most noticeable. Both tables are listed in Appendix B, *Character Sets.*

The Engine can be instructed to use an alternate translation table - the **TABLE** option is used to specify this table. The translation information is located in a specially formatted file, provided by Telamon. Two such files are found in the **NETWORKS** group of the **TELAMON** account:

```
AEHP.NETWORKS.TELAMON
```

and:

```
AEIBM.NETWORKS.TELAMON
```

These two files appear as tables appear in Appendix B. The **AEHP** table is the default table used by the Engine - it need not be specified as its values are permanently loaded in the Engine's firmware. The **AEIBM** table contains the values matching those found in the *IBM 3780* manuals. To specify this table, enter:

```
#RJLINE ... ;TABLE=AEIBM.NETWORKS.TELAMON
```

or:

```
TABLE = AEIBM.NETWORKS.TELAMON
```

in the **RJECLINE** file.

Don't attempt to modify these files. Network/S performs a number of internal consistency checks on the data prior to downloading the table to the Engine - if the format isn't valid, the program will terminate with an error condition. If different translations are required, contact Telamon and we will try to create a suitable translation table for you.

Syntax:

```
#RJLINE ... ;TABLE={filename}
```

Default:

Default HP ASCII-to-EBCDIC table is used.

RTSCTS

Specifies an RTS-CTS turnaround delay to be used in conjunction with that provided by the modem. If a non-zero value is specified, the Network Engine will delay between raising the RTS signal and waiting for CTS the indicated amount of time, expressed in milliseconds. This ensures that at least this much time will pass before any information is transmitted.

Syntax:

```
#RJLINE ... ;RTSCTS=milliseconds
```

Default:

```
#RJLINE ... ;RTSCTS=0
```

Example

To make a connection to a remote host using the 3780 protocol with EBCDIC control codes, dialing 555-1234 using a SADL-compatible modem, with a default maximum block size of 10 records and/or 512 bytes, using an Engine attached to logical device **49**, enter:

```
#RJLINE 3780;LINECODE=EBCDIC; &
#  CONNECT=DIAL,"5551234",SADL;MAXRPB=10,512;DEV=49
```

To make a connection answering an incoming 2780 call with ASCII control codes on device **49**, enter:

```
#RJLINE 2780;LINECODE=ASCII;CONNECT=ANSWER;DEV=49
```

Note that except for opening the port and verifying the Engine's existence, the **#RJLINE** command is not acted upon until the first **#RJIN, #RJOUT** or **#RJLIST** command that follows is executed or the **NOW** parameter is used on the **#RJLINE** command.

To set up a connection on a direct-connect line for 3780 using an Engine connected to a port configured with device class **RJLINE** and all other parameters set to their default values, enter:

```
#RJLINE 3780
```

RJMPE

Execute MPE commands

The **#RJMPE** command is used to execute MPE commands. e.g.

```
#RJMPE PURGE DATAFILE
#RJMPE FILE DATAFILE;REC=-128,2,F,ASCII
#RJOUT *DATAFILE
```

and:

```
#RJMPE RUN FCOPY.PUB.SYS;INFO="FROM=DBFILE;TO=RJINFILE;NEW"
```

Syntax:

```
#RJMPE command
```

RJOUT/RJLIST/RJPUNCH

Receive a File from a Remote Host

The **#RJOUT**, **#RJLIST**, and **#RJPUNCH** commands are used to initiate a data transfer of output from the remote host system to the local computer. The following table summarizes the type of data each command will accept:

TABLE 3-2.

Command	Data To Be Received		
	Unrouted	Routed to List	Routed to Punch
#RJOUT	Accept	Accept	Accept
#RJLIST	Accept	Accept	Error
#RJPUNCH	Accept	Error	Accept

Unless you are receiving routed list files and you need to take advantage of **#RJLIST** parameters not available with **#RJOUT**, **#RJOUT** will usually suffice for all receive operations.

In the examples that follow, **#RJOUT** is used. **#RJLIST** and **#RJPUNCH** can be used instead when indicated.

The format of the command is:

```
#RJOUT [Data Destination] [;option] [;option] ...
```

If the **#RJOUT** command has been preceded by an **#RJIN** or **#RJBID**, Network will execute the equivalent of the **#RJEOD** command before proceeding with the **#RJOUT** command. This is necessary in order to transition from the Transmit state to the Control state, prior to entering the Receive state.

The parameters to **#RJOUT** include:

JCW, Variable and Indirect File substitution will be performed on the text of the command prior to its evaluation. JCWs and MPE/iX variables are identified by a "!" prefix, Variables are identified by a "%" prefix and Indirect Files are identified by a "^" prefix. When an Indirect File is specified, the first record of the file will be read and substituted for the filename in the command. For example, to substitute the variables "M" and "S" in the **WAIT** option, enter:

```
#RJCALC %M = 2
#RJCALC %S = 30
#RJOUT filename;WAIT=%M,%S
```

This evaluates to:

```
#RJOUT filename;WAIT=2,30
```

Data Destination

The **#RJOUT** command can specify that data be sent to a file or to an SL/XL interface procedure in the following form:

```
#RJOUT [ filename |
         "device name" |
         device number |
         @procedurename [ (lib) ]
       ] [,count]
```

If 'filename' is specified, it must be a valid MPE filename.

"Device name" or *device number* must refer to configured devices on your HP 3000. The files opened using either of these two specifications will be unnamed.

If no target is specified, unrouted data will be written to the file associated with **RJELIST**, which is **$STDLIST** by default. Routed data will be directed to the appropriate **RJELIST** or **RJEPUNCH** file, as dictated by the routing information.

If ,*count* is specified, the **#RJOUT** command will be executed up to *count* times consecutively, concatenating the received data sets. The ,*count* parameter will also serve to render any **WAIT** time-out that may occur as a soft error - Network/S will continue with the next command in sequence.

By default, Network/S first attempts to open the output 'filename' as an old, permanent file. If the file doesn't exist, the file is opened as a new file. The default characteristics for the new file are, in :FILE equation format:

```
:FILE filename,NEW;REC=,,V,ASCII;CCTL;ACC=APPEND;EXC
```

Note that **CCTL** is applied to the default setting! To receive text files without carriage control it will be necessary either to issue a file equation specifying the **NOCCTL** option, to build the file without **CCTL** or to specify the **CCTL=NO** option in the **#RJOUT** command.

See Appendix D, *Procedure Examples*, for an example **#RJOUT** procedure.

RJOUT Procedure

The **#RJOUT** command normally directs its output to a file identified by the item located between the **#RJOUT** command and the first semi-colon (;).

An optional technique involves the use of a user-written library procedure, located in an SL (Compatibility Mode) or an XL (Native Mode.)

When specifying an external procedure, the syntax of the **#RJOUT** command is:

```
#RJOUT @procedurename [ ( lib ) ]
```

Network/S will attempt to load the specified procedure, in the exact specified case, from the SL or XL file specified by *lib.*

When running Network/S on an MPE/V based HP 3000 or in Compatibility Mode on an MPE/XL or MPE/iX based HP 3000, *Lib* can be one of the following:

Lib	SL Search sequence	
---	---	
G	SL.*logongroup*.*logonaccount*	then
	SL.PUB.*logonaccount*	then
	SL.PUB.SYS	
P	SL.PUB.*logonaccount*	then
	SL.PUB.SYS	
S	SL.PUB.SYS only	
GX	SL.*appgroup*.*appaccount*	then
	SL.PUB.*appaccount*	then
	SL.PUB.SYS	

PX	SL.PUB.*appaccount*	then
	SL.PUB.SYS	

where *logongroup* and *logonaccount* refer to the user's logon group and account and *appgroup* and *appaccount* refer to the location where the Network/S application resides, by default, NETWORKS and TELAMON respectively. Note that if Network/S has been placed in PUB.SYS, the only meaningful options for *Lib* are G, P and S since GX and PX would both refer to PUB.SYS.

When running the Native Mode version of Network/S on an MPE/iX based HP 3000, Network/S will attempt to load the specified procedure from the XL file identified by *lib*, which must be a valid filename.

See Appendix D, *Procedure Examples*, for an example **#RJOUT** procedure.

LIST

When **#RJOUT** is used to receive routed output, routed List data is normally written to the default list file, specified when Network/S was started. All subsequent routed List data sets are then written to this one file.

The **LIST** option provides the means to specify a different routed List destination for each **#RJOUT** command.

If Network/S builds this file, it is created with the following characteristics:

```
REC=-256,,V,ASCII;CCTL;DISC=5000
```

Syntax:

#**RJOUT ... ;LIST=*filename***

Default:

Routed List data sets directed to the **RJELIST** file

PUNCH

When **#RJOUT** is used to receive routed output, routed Punch data is normally written to the default punch file, specified when Network/S was started. All subsequent routed punch data sets are then written to this one file.

The **PUNCH** option provides the means to specify a different routed punch destination for each **#RJOUT** command.

If Network/S builds this file, it is created with the following characteristics:

```
REC=40,,F,BINARY;NOCCTL;CODE=RJEPN
```

Syntax:

#**RJOUT ... ;PUNCH=*filename***

Default:

Routed punch data sets directed to the **RJEPUNCH** file

OUTCODE

The **OUTCODE** parameter is used to identify the desired character set for the data to be received by this **#RJOUT** command. The Engine uses this information the translate the data, if necessary, as it is received. The translation is performed according to the **#RJLINE ... ;LINECODE** setting. The possible values are:

ASCII	The data is ASCII, and is converted if **LINECODE=EBCDIC**
EBCDIC	The data is EBCDIC, and is converted if **LINECODE=ASCII**
BINARY	The data is not to be converted after reception

Warning: If the file to be received contains packed-decimal data, use **OUTCODE=BINARY** - otherwise, Network/S will search for and convert apparent carriage control directives and blank-compression codes, possibly corrupting the data.

Syntax:

#**RJOUT ... ;OUTCODE={ASCII | EBCDIC | BINARY}**

Default:

#**RJOUT ... ;OUTCODE=ASCII**

OUTSIZE

The **OUTSIZE** parameter is used to specify the maximum amount of data to be written to the Data Destination for each record received. Normally, this value defaults to the logical record width of the specified data file. If a logical record is received with fewer characters than indicated by **OUTSIZE**, the record will be filled with either ASCII or EBCDIC blanks or binary zeroes, according to the code specified by **OUTCODE**. If too many characters are received, the overflow will be discarded if **TRUNCATE=YES** is selected; otherwise Network/S will attempt to deblock the received data, **OUTSIZE** bytes or words at a time. For example, if five 80-byte records are to be received in each block, for a total of 400 bytes per block, the following will cause Network/S to correctly deblock the records:

#**RJOUT ... ;OUTSIZE=-80,5**

OUTSIZE=-80,5 will cause Network/S to inform the Engine that 400 byte blocks are to be expected. Network/S will deblock the received data 80 bytes at a time - if, for instance, only 240 bytes are received, Network/S will write three 80-byte records to the output file.

Please note:

- If **TRUNCATE=YES** is selected on the **#RJOUT** command, this automatic deblocking of transparently blocked data will not be performed. In the above example, only the first 80 bytes per block would be written to the file.
- If the file opened by the **#RJOUT** command has a record width different from that specified by the **OUTSIZE** parameter, the smaller of the two values will be used. **OUTSIZE** is *not* used when opening/building the specified **#RJOUT** file - use a **:FILE** equation to specify the record width for a new file.
- If non-transparent received records are blocked with separator characters (**RS**/**US**/**EM**), the Engine will automatically deblock the records on those boundaries. **OUTSIZE**, if specified, will only be used to pad short records.

Syntax:

```
#RJOUT ... ;OUTSIZE=[{-bytes | +words}] [,max records per block]
```

Default:

```
#RJOUT oldfile;OUTSIZE={output file's record width}
#RJOUT newfile;OUTSIZE=-256
#RJOUT @procedurename;OUTSIZE=-80
```

TRUNCATE

The **TRUNCATE** parameter is used to specify what Network/S is to do with received data records whose width exceeds the record width of the output file. If set to **YES**, Network/S will automatically discard the excess data. If set to **NO**, Network/S will write as many records to the file as necessary to completely output the entire received record.

Syntax:

```
#RJOUT ... ;TRUNCATE={YES | NO}
```

Default:

```
#RJOUT ... ;TRUNCATE=NO
```

WAIT

In bi-sync communications, whenever a sender wishes to initiate transmission of a data set, it must first indicate to the remote system its intention to do so. This is referred to as 'bidding' for the line.

The **WAIT** parameter is used to specify how long Network/S will wait for a bid request to be received. If no bid is received in the specified time interval Network/S will return an error indicating that the specified file could not be received. **WAIT** also specifies the maximum time interval between successive data blocks - in effect, an idle timer.

If the ',count' parameter is specified on the **#RJOUT** command and the **WAIT** timer expires while Network/S is waiting *between* files, no error is generated - Network/S continues with the next command.

Specifying **WAIT=0** will cause Network/S to wait indefinitely.

Syntax:

```
#RJOUT ... ;WAIT=[minutes] [,seconds]
```

Default:

```
#RJOUT ... ;WAIT=3,0
```

CCTL

By default, when Network/S creates the destination **#RJOUT** output file, the MPE option CCTL is specified. As a direct result, any received carriage control directives will be converted to their MPE equivalents before data is written to this file.

To override this feature, you can do one of the following:

- Build the output file (via :BUILD) without specifying the CCTL option
- Issue a **:FILE** equation for a new output file specifying the **NOCCTL** option
- Include the **CCTL=NO** option in the **#RJOUT** command

If **CCTL=NO** or **CCTL=SKIP** is specified, Network/S will build the new output file (if it doesn't already exist) with **NOCCTL** specified. With **CCTL=NO**, any IBM-style carriage control directives found in the data stream will be deleted before the data is written to the file. Note that if the specified output file already exists and that it has been built using the CCTL option, Network/S will have to use the existing file, regardless of the **#RJOUT CCTL** setting. **CCTL=NO** will still cause control codes to be intercepted - in this case, however, they will be ignored. Use **CCTL=SKIP** if you expect to receive data with Escape codes ($1B-ASCII, $27-EBCDIC) in the first position of any record that might be mistaken for carriage control.

Conversely, if **CCTL=YES** is specified and an existing MPE file is found built with the **NOCCTL** option, Network/S will behave as though the **CCTL=NO** option had been specified.

Thus, Network/S will correctly convert an IBM-style print file to an MPE-style file if, and only if, both **CCTL=YES** is specified in the **#RJOUT** command and **CCTL** is specified in the output file's format.

If the file to be received already contains HP-style carriage control codes, use **CCTL=HP** to direct Network/S to use these codes, “as-is”.

Syntax:

```
#RJOUT ... ;CCTL={YES | NO | HP | SKIP}
```

Default:

```
#RJOUT ... ;CCTL=YES
```

OLDFF

Some versions of the MPE/iX operating system have demonstrated a problem handling the %300 Skip to Channel 1 form feed carriage control code. The **OLDFF** keyword provides the means to force Network/S to use the %061 form feed code, which performs correctly under these circumstances.

Syntax:

```
#RJOUT ... ;OLDFF={YES | NO}
```

Default:

```
#RJOUT ... ;OLDFF=YES(MPE/iX), NO(MPE/V)
```

SPACING

A file created by **#RJOUT** can be setup for either Pre-spacing or Post-spacing. The usual system default is to perform Post-spacing - that is, write text then perform carriage control.

If **SPACING=PRE** is selected, Pre-spacing will be set in the file, with carriage control operations performed *prior* to write operations.

If **SPACING=POST** is selected, Post-spacing will be set, with carriage control operations performed *after* write operations.

If **SPACING=NONE** is selected, Network/S will write received text to the file without inserting any extra carriage control codes.

Syntax:

```
#RJOUT ... ;SPACING={PRE | POST | NONE}
```

Default:

```
#RJOUT ... ;SPACING=NONE
```

REPEAT

The **REPEAT** keyword provides the means to cause the specified **#RJOUT** command to be repeated indefinitely. This is usually applied with the Programmatic Access mode of operation.

If **REPEAT=YES** is specified, the **#RJOUT** command will be repeated indefinitely, until either a programmatic interrupt occurs specifying a new **#RJOUT** command with **REPEAT=NO** or terminating Network/S, or until Control-Y is entered when Network/S is run interactively.

If **REPEAT=NO** is specified, repeat mode is explicitly terminated.

If the **REPEAT** keyword is omitted, the current repeat state of an interrupted **#RJOUT** command (if any) is left unchanged.

See Appendix F, *Programmatic Access*, for more information on its use.

Syntax:

```
#RJOUT ... ;REPEAT={YES | NO}
```

Default:

```
#RJOUT ... ;REPEAT=NO
```

INTERRUPT

The **INTERRUPT** keyword is used to indicate whether the current **#RJOUT** command can be interrupted via the **#RJLINE MSGFILE** interrupt file.

If **INTERRUPT=YES** is specified, the **#RJOUT** command is executed normally, but can be interrupted if a command request appears in the **MSGFILE** file before a transmit bid sequence is received from the remote system.

If **INTERRUPT=NO** is specified, no such interrupt can occur.

See Appendix F, *Programmatic Access*, for more information on its use.

Syntax:

```
#RJOUT ... ;INTERRUPT={YES | NO}
```

Default:

```
#RJOUT ... ;INTERRUPT=NO
```

EMPTYOK

Normally, if an entirely empty data set is received during the execution of the **#RJOUT** command, that set is ignored, and the **#RJOUT** command continues waiting for valid data. If **EMPTYOK** is set to **YES**, empty data sets are to be treated as valid, albeit empty, files, and Network/S will move on to the next command to be executed.

Similarly, if an empty (zero length) block is received at the start of a transmission, it is normally ignored. Setting **EMPTYOK** to **YES** will cause the empty block to be processed as valid data.

Note that some host systems send the sequence of a line bid followed immediately by an **EOT** as a “busy-wait” mechanism. The intent here is to provide some activity every few seconds to prevent the remote site (Network/S in this case) from timing out. Don't use **EMPTYOK** in this instance. Network/S will incorrectly move on to the next command, when, in fact, the host hasn't begun to send its data.

Syntax:

```
#RJOUT ... ;EMPTYOK={YES | NO}
```

Default:

```
#RJOUT ... ;EMPTYOK=NO
```

REBLOCK

When Network/S receives data blocks, it attempts to deblock transparent data based upon the **OUTSIZE** parameter and non-transparent data based upon the presence of record delimiter characters (**RS**, **US**, **EM**, **NL**.) By default, if any data block is received that is not terminated by one of these characters, that is, a record ends on the normal **ETB**/**ETX** block termination character, that record is processed as though it had been correctly terminated. Similarly, if there is insufficient data to fill an integral number of transparent records, the partial (short) portion is processed normally. This is done to maintain compatibility with Hewlett-Packard's RJE product.

If **REBLOCK** is set to **YES**, Network/S will save any such partial records until subsequent data is received with the correct terminator or until sufficient data is received to fill the next transparent data record. In this fashion, Network/S can correctly reconstruct data sent by another system that *can* block partial records into the transmission block. (See **#RJIN REBLOCK** (page 3-40) for information on how Network/S can send data in this fashion.)

If **REBLOCK** is set to **LENGTH**, Network/S will deblock received data solely based upon the record width of the output file. This provides a convenient method for converting variable length records into a fixed length format on the HP 3000. For example, if you are receiving variable width EDI X.12 records, but your translation software requires that the data be packed into an 80-byte record format, you could enter:

```
#RJMPE BUILD EDIDATA;REC=-80,16,F,ASCII
#RJOUT EDIDATA;REBLOCK=LENGTH
```

Network/S will write records to **EDIDATA** only when at least 80 bytes have been received. If the final record to be written is less than 80 bytes, it will be padded with blanks or zeroes, according to the **OUTCODE** specified.

As many X.12 formats use the EBCDIC New-Line character (**NL**) as a segment delimiter, you would probably also specify **NL=NO** on the above **#RJOUT** line.

Syntax:

```
#RJOUT ... ;REBLOCK={YES | NO | LENGTH}
```

Default:

```
#RJOUT ... ;REBLOCK=NO
```

TIMEOUTOK

Normally, when the specified **WAIT** time interval passes during the execution of an **#RJOUT** command, an error (CSERR 217) results, causing Network/S to terminate. The **TIMEOUTOK** option, when set to **YES**, will cause any such time-out to be treated as a successful termination to the **#RJOUT** command and Network/S will continue with the next command.

Set **TIMEOUTOK** to **ETX** to cause a timeout that occurs after the receipt of an ETX-terminated block not to be considered an error.

Syntax:

```
#RJOUT ... ;TIMEOUTOK={YES | NO | ETX}
```

Default:

```
#RJOUT ... ;TIMEOUTOK=NO
```

HT

Received data streams are examined for the presence of horizontal tab characters (**HT**, $09-ASCII, $05-EBCDIC.) If a tab-definition record has been previously received, the **HT** characters will be replaced with a suitable number of blanks to skip to the next defined tab position. If no tab-definition record has been received, the **HT** characters will be replaced with spaces. (Tab-definition records appear at the beginning of a received file and start with an **ESC**-**HT**.)

Under certain circumstances, it would be desirable to disable this automatic transformation. Setting **HT=NO** will suppress the conversion of horizontal tab characters for the current file transfer only. Setting **HT=NO** in the **#RJLINE** command will, by default, affect all **#RJOUT**s.

Syntax:

```
#RJOUT ... ;HT={YES | NO}
```

Default:

```
#RJOUT ... ;HT=value of RJLINE HT setting
```

FF

Received 3780 data streams are examined for the presence of the form-feed character (**FF**, $0C-ASCII/EBCDIC.) These are treated as a record delimiters, causing any preceding data to be written to disc as a logical record, with suitable carriage control.

Under certain circumstances, it would be desirable to disable this feature. Setting **FF=NO** will suppress this interpretation of form-feed characters for the current file transfer only. Setting **FF=NO** in the **#RJLINE** command will, by default, affect all **#RJOUT**s.

Syntax:

```
#RJOUT ... ;FF={YES | NO}
```

Default:

```
#RJOUT ... ;FF=value of RJLINE FF setting
```

LF

Received 3780 data streams are examined for the presence of the line-feed character (**LF**, $0A-ASCII, $25-EBCDIC.) These are treated as a record delimiters causing any preceding data to be written to disc as a logical record with suitable carriage control.

Under certain circumstances, it would be desirable to disable this feature. Setting **LF=NO** will suppress this interpretation of line-feed characters for the current file transfer only. Setting **LF=NO** in the **#RJLINE** command will, by default, affect all **#RJOUT**s.

Syntax:

```
#RJOUT ... ;LF={YES | NO}
```

Default:

```
#RJOUT ... ;LF=value of RJLINE LF setting
```

NL

Received 3780 data streams are examined for the presence of the New-Line character (**NL**, $85-ASCII, $15-EBCDIC.) These are treated as a record delimiters causing any preceding data to be written to disc as a logical record with suitable carriage control.

Under certain circumstances, it would be desirable to disable this feature. Many EDI systems, for example, use the **NL** character as an integral part of the data record. Splitting the record at this point renders the data unusable. Setting **NL=NO** will suppress this interpretation of new-line characters for the current file transfer only. Setting **NL=NO** in the **#RJLINE** command will, by default, affect all **#RJOUT**s.

Syntax:

```
#RJOUT ... ;NL={YES | NO}
```

Default:

```
#RJOUT ... ;NL=value of RJLINE NL setting
```

VT

Received 3780 data streams are examined for the presence of the vertical-tab character (**VT**, $0B-ASCII/EBCDIC.) These are treated as a record delimiters causing any preceding data to be written to disc as a logical record with suitable carriage control.

Under certain circumstances, it would be desirable to disable this feature. Setting **VT=NO** will suppress this interpretation of vertical-tab characters for the current file transfer only. Setting **VT=NO** in the **#RJLINE** command will, by default, affect all **#RJOUT**s.

Syntax:

```
#RJOUT ... ;VT={YES | NO}
```

Default:

```
#RJOUT ... ;VT=value of RJLINE VT setting
```

ECHO

When set to **YES**, the **ECHO** option will cause each record written to disc to be displayed, in addition, upon **$STDLIST**. If the output file is **$STDLIST**, this option will be ignored.

Syntax:

```
#RJOUT ... ;ECHO={YES | NO}
```

Default:

```
#RJOUT ... ;ECHO=NO
```

AUTONUM

Normally, when a **#RJOUT *file,count*** command has been issued, up to *count* data sets will be received and written to the one *file*, with each data set in effect appended to the one that preceded it. No marker is provided to indicate where one data set ends and the next data set begins.

The **AUTONUM** option provides a means for instructing Network/S to close the output file after each data set has been received and to automatically create a new file for the next data set. Each subsequent file is built with characteristics matching those of the first file specified, except that the filename is modified to contain a unique numeric field, which is increased by one for each data set received.

In order for this option to work properly, the filename specified in the **#RJOUT** command *must* contain at least one numeric digit. Network/S will identify a field in the filename, starting at the first numeric digit found and extending through the last contiguous numeric digit. For example, if the command specified is:

```
#RJOUT TEXT0001,3;AUTONUM=YES
```

Network/S will use the last four bytes of the filename to uniquely identify each data set, with the first data set received called **TEXT0001**, the second data set will be called **TEXT0002**, the third data set will be called **TEXT0003**, and so on. Each file built after **TEXT0001** will be opened with characteristics matching those of **TEXT0001**.

If the command specified is:

```
#RJOUT ABC100RP,5;AUTONUM=YES
```

Network/S will use the fourth through sixth bytes of the filename to uniquely identify each data set. The first data set received will be called **ABC100RP**, the second data set will be called **ABC101RP**, the third data set will be called **ABC102RP**, and so on. Note that the numbering scheme uses the specified numeric field as its starting value.

If no numeric field is found in the original filename, the **AUTONUM** option will be ignored. If the numeric field proves too short for the number of files received, the number will overflow to the left, replacing alphabetic characters that precede the numeric field. Take care in specifying the location for the numeric field. MPE filenames must start with an alphabetic character.

Syntax:

```
#RJOUT ... ;AUTONUM={YES | NO}
```

Default:

```
#RJOUT ... ;AUTONUM=NO
```

ETX

The **ETX** keyword will cause the **#RJOUT** command to treat the receipt of an End-of-Text (**ETX**) terminated data block in the same manner as the receipt of an End-of-Transmission (**EOT**) code. The current file will be closed and processing will continue normally.

This option should be used primarily in conjunction with the **AUTONUM** option, which will allow for the creation of unique filenames for each data set received.

If not used with **AUTONUM**, you will have to issue the appropriate number of **#RJOUT** commands to accommodate the expected number of incoming data sets. The **EOTRECEIVED** JCW will be set to one (1) when the **#RJOUT** command has received an **EOT**; it will be set to zero (0) and the **ETXRECEIVED** JCW will be set to one (1) when the **#RJOUT** command has terminated due to the receipt of an **ETX**-terminated block.

Syntax

```
#RJOUT ... ;ETX={YES | NO}
```

Default:

```
#RJOUT ... ;ETX=NO
```

NEEDETX

Most bi-sync hosts follow the practice of terminating all but the last data block with an End-of-Block (**ETB**) character, with the last block terminated with an End-of-Text (**ETX**) character. If no more data is pending, the host then sends an End-of-Transmission (**EOT**) which terminates the **#RJOUT** command.

Network/S can test for the presence of the **ETX** terminator and either generate a warning, if **#RJLINE WARN** is set, or an error (CSERR 220), if **NEEDETX** is set.

While some host systems regularly send files with no **ETX** terminator, it is an unusual practice. Very often, the fact that the **#RJOUT** command has finished without having seen the **ETX** indicates that the file may not have been sent successfully and/or in its entirety.

Set **NEEDETX** to **YES** if the host system's specifications indicate that all transfers will include a final block with an **ETX** terminator. Then, if an **EOT** is received and Network/S hasn't seen the **ETX** terminated block, an error condition will be generated.

You may also want to set **NEEDETX** if the **XEND** option has been specified on the **#RJLINE** command. Since **XEND** causes received **DLE-EOT** disconnect sequences to be treated as **EOT**s, an abnormally terminated transmission might not otherwise be noticed!

Syntax

```
#RJOUT ... ;NEEDETX={YES | NO}
```

Default:

```
#RJOUT ... ;NEEDETX=NO
```

MARKETX

The **MARKETX** keyword will cause the **#RJOUT** command to insert a line into the output file whenever an End-of-Text (**ETX**) terminated block has been received. **ETX** terminated blocks mark individual data sets.

When **MARKETX** is set, the following text line will be written to the output destination:

```
----------Network/S: End of Data-set----------
```

This option should be used if the remote system sends multiple data sets and there is no easy means for determining the logical break between data sets once they have arrived. An alternate scheme involves the use of the **ETX** option, which will cause individual data sets to be written to separate files.

Syntax:

```
#RJOUT ... ;MARKETX={YES | NO}
```

Default:

```
#RJOUT ... ;MARKETX=NO
```

AUTOPAGE

If it is expected that the received printer output contains embedded vertical forms control (VFC) codes, set **AUTOPAGE=NO**. Otherwise use **AUTOPAGE=YES**.

When **YES** is selected, Network/S will use carriage control codes that provide for automatic page ejects every 60 lines. When **NO** is selected, it assumed that the control codes in the received data will provide pagination.

Syntax:

```
#RJLIST ... ;AUTOPAGE={YES | NO}
```

Default:

```
#RJLIST ... ;AUTOPAGE=NO
```

FORMSMSG

The **FORMSMSG** parameter allows the specification of a forms message that will be displayed on the System Console whenever the received, spooled output of the **#RJLIST** command is ready to be printed.

See the description for the **FORMS** keyword on the **:FILE** command for more information.

Syntax:

```
#RJLIST ... ;FORMSMSG=forms message[.]
```

Default:

no forms message

Routing

The **#RJOUT** command initiates the receipt of one or more data sets from the sending system. This reception is initiated when Network/S receives an incoming line bid and finishes when the sending system transmits an End-of-Transmission (**EOT**) code.

Each block in the transmission consists of one or more records from the source file and each but the last block is terminated with an End-of-Text-Block (**ETB**) code. The last block in the transmission is terminated with an End-of-Text (**ETX**) code, usually immediately followed by the **EOT**.

If more than one data set is to be sent during this transmission, the last block in each data set will be terminated with an **ETX**, and will then be followed by another data set, not an **EOT**. By default, the **#RJOUT** command will treat the multiple data sets as a single data stream, combining everything into a single file on the local computer.

Each data set may optionally begin with a routing code, also known as a device select code. The routing code is used to indicate whether the data to follow is to be treated as line printer (list) output or punch output. Remember that IBM 2780 and 3780 terminals can be equipped with a line printer and one or two card punches. The routing codes are necessary to "point" the data stream to the appropriate device.

The routing codes aren't required: if no code is present, the data set is treated as list data.

#RJLIST can process routed list output and unrouted output. **#RJPUNCH** can process routed punch output and unrouted output. **#RJOUT** can process both routed and unrouted output, and if a file or procedure is specified in the **#RJOUT** command, all data will be sent to that destination, regardless of the routing.

Only when no file has been specified in the **#RJOUT** command will routing take place. Under these circumstances, routed punch output will be directed to the default punch file, and routed list output and unrouted output will be directed to the default list file, specified in the **:NERJE** command, or by way of the **LIST** and/or **PUNCH** options of the **#RJOUT** command.

The default list file is **$STDLIST** and the default punch file is **$NEWPASS**, both of which can be changed in the **:NERJE** command. For example:

```
:FILE LIST;DEV=LP
:FILE PUNCH,NEW;SAVE;DEV=DISC
:NERJE ,,*LIST,*PUNCH
#RJLINE ...
#RJIN logon
#RJOUT
#RJEND
```

RJLIST Example

To receive a unrouted print file to be copied to the system line printer (LP), enter:

```
#RJLIST *LP
```

If the file equation for LP has CCTL specified, either explicitly or implicitly, and the received text contains valid IBM-style printer control codes, those codes will be converted according to the following table:

TABLE 3-3.

Emulation		LINECODE		Carriage Control Function	AUTOPAGE	
2780	3780	ASCII	EBCDIC		YES	NO
*	*	ESC Q	ESC /	Single Space	%040	%201
*	*	ESC R	ESC S	Double Space	%060	%202
*	*	ESC S	ESC T	Triple Space	%304	%203
*	*	ESC A	ESC A	Skip to Channel 1 (1)	%300	%300
*	*	ESC B	ESC B	Skip to Channel 2	%301	%301
*	*	ESC C	ESC C	Skip to Channel 3	%302	%302
*	*	ESC D	ESC D	Skip to Channel 4	%303	%303
*	*	ESC E	ESC E	Skip to Channel 5	%304	%304
*	*	ESC F	ESC F	Skip to Channel 6	%305	%305
*	*	ESC G	ESC G	Skip to Channel 7	%306	%306
*	*	ESC H	ESC H	Skip to Channel 8	%307	%307
	*	ESC I	ESC I	Skip to Channel 9	%310	%310
	*	ESC J	ESC J	Skip to Channel 10	%311	%311
	*	ESC K	ESC K	Skip to Channel 11	%312	%312
	*	ESC L	ESC L	Skip to Channel 12	%313	%313
	*	ESC M	ESC M	Suppress Space	%053	%053

Note: (1) - Network/S may optionally use %061 as the form-feed code. See the **#RJOUT/#RJLIST OLDFF** keyword for more information.

RJOUT Example

Example 1:

To receive an 80-byte ASCII text file, named **DATA**, from a remote host, enter:

```
#RJOUT DATA;OUTSIZE=-80
```

Note the following:

- By default, **DATA** will be built with **CCTL**.
- Network/S will automatically determine if the data has been sent in transparent mode or with compression enabled.
- The **OUTSIZE** parameter will only be utilized if the data has been sent in transparent mode. Otherwise, embedded ITB characters will indicate the record size.

- If the output data file has a record width of 80 bytes, the **OUTSIZE** keyword does not have to be specified. Network/S uses the disc record width as its default.

Example 2:

To receive an HP 3000 program file, named **PROGRAM**, from another HP 3000 running either HP's RJE or Network/S, enter:

```
#RJMPE FILE PROGRAM;CODE=PROG;&
# REC=128,1,F,BINARY;NOCCTL;DISC=,1
#RJOUT PROGRAM;OUTCODE=BINARY;OUTSIZE=128
```

Note the following:

- The **OUTSIZE** parameter may be omitted since Network/S uses the file's record width as its default value.
- With **OUTCODE=BINARY**, no compression or truncation will be expected or acted upon.

Example 3:

Some EDI Value-Added Networks send you a variable number of files when you send a request to receive the contents of your mailbox. Using combinations of Network/S variables, it is possible to receive the files either as a single file on your system or as multiple, individual files.

To receive multiple files as a single file, enter:

```
#RJOUT MAILBOX,10;WAIT=0,10;EMPTYOK=YES;...
```

Note the following:

- The count parameter of 10 will receive up to 10 separate file transmissions (line bids).
- The **WAIT=0,10** parameter will wait for 10 seconds to receive the first, and each subsequent, file up to the 10th file. Experiment to determine a suitable **WAIT** value. If the value is too short, the **#RJOUT** command may terminate prematurely. As a result, you may not receive all the messages in your EDI VAN mailbox. On the other hand, a value that is too long may cause the EDI VAN to time-out and drop the line, assuming that Network/S has stopped responding. This would cause Network/S to abort on the next command unless it is an **#RJEND**.
- Since a count has been specified in the **#RJOUT** command, when and if Network/S times out awaiting a line bid, the program will continue processing with the next command in the command file. Only if a time-out occurs *during* the receipt of any data set will Network/S generate an error. For example, if there are 6 messages in the EDI VAN mailbox and the EDI VAN does not send a linebid within 10 seconds while Network/S is waiting for a 7th message, Network/S will time-out on the **#RJOUT** command but will continue and process the next command. A line drop (CSERR 103) prior to a bid will also be ignored under the above circumstances.
- The **EMPTYOK=YES** parameter allows Network/S to continue processing the next command even if it receives an empty file (a line bid followed by an **EOT**). Some systems send an “empty” file as an indication that no more files remain to be sent.
- Up to 10 files in the EDI VAN mailbox will be written to the file **MAILBOX** on your system.

If you want to receive each file from the EDI VAN separately, add the **AUTONUM=YES** parameter to the **#RJOUT** command.

```
#RJOUT MAIL01,10;AUTONUM=YES;WAIT=0,10;EMPTYOK=YES
```

The **AUTONUM** parameter will cause Network/S to create a separate file for each file received from the EDI VAN. See the **AUTONUM** parameter of the **#RJOUT** (page 3-88) command for more information.

RJPAUSE

Pause during Program Execution

The **#RJPAUSE** command provides the means to pause during the execution of Network/S. If no parameter is provided, **#RJPAUSE** will pause for one second; otherwise, it will pause for the specified number of seconds.

Care should be taken while executing the **#RJPAUSE** command. While Network/S is paused, no Engine activity will be observed, and more importantly, the Engine will observe no activity from Network/S. If the Engine's Crash timer is set and if the pause interval exceeds the Engine's Crash timer value, the Engine will reset and drop the line.

JCW, Variable and Indirect File substitution will be performed on the text of the command prior to its evaluation. JCWs and MPE/iX variables are identified by a "!" prefix, Variables are identified by a "%" prefix and Indirect Files are identified by a "^" prefix. When an Indirect File is specified, the first record of the file will be read and substituted for the filename in the command. For example, to substitute the variable "S" below, enter:

```
#RJCALC %S = 60
#RJPAUSE %S
```

This evaluates to:

```
#RJPAUSE 60
```

Syntax:

```
#RJPAUSE [n]
```

RJSHOWTIME

Display Current Date/Time

The **#RJSHOWTIME** command displays a time-stamp in the form:

ddMMMyy-hh:mm:ss.t

Syntax:

#**RJSHOWTIME**

RJSTAT

Display Transfer Statistics

The **#RJSTAT** command displays summary statistics.

The format of the command is:

```
#RJSTAT [filename | { @procedurename [ (lib) ] } ] [;RESET]
```

If no target is specified, data will be sent to **$STDLIST**, or the file associated with **RJELIST**.

If **RESET** is specified, all counters will be reset to zero after the **#RJSTAT** display is generated.

JCW, Variable and Indirect File substitution will be performed on the text of the command prior to its evaluation. JCWs and MPE/iX variables are identified by a "!" prefix, Variables are identified by a "%" prefix and Indirect Files are identified by a "^" prefix. When an Indirect File is specified, the first record of the file will be read and substituted for the filename in the command. For example, to substitute the variable "N" in the filename, enter:

```
#RJCALC %N = 100
#RJSTAT STAT%N
```

This evaluates to:

```
#RJSTAT STAT100
```

RJSTAT Procedure

The **#RJSTAT** command normally directs its output to a file identified by the item immediately following the **#RJSTAT** command.

An optional technique involves the use of a user-written library procedure, located in an SL (Compatibility Mode) or an XL (Native Mode.)

When specifying an external procedure, the syntax of the **#RJSTAT** command is:

```
#RJSTAT @procedurename [ ( lib ) ]
```

Network/S will attempt to load the specified procedure, in the exact specified case, from the SL or XL file specified by *lib.*

When running Network/S on an MPE/V based HP 3000 or in Compatibility Mode on an MPE/XL or MPE/iX based HP 3000, *Lib* can be one of the following:

Lib	SL Search sequence	
---	---	
G	SL.*logongroup*.*logonaccount*	then
	SL.PUB.*logonaccount*	then

	SL.PUB.SYS	
P	SL.PUB.*logonaccount*	then
	SL.PUB.SYS	
S	SL.PUB.SYS only	
GX	SL.*appgroup*.*appaccount*	then
	SL.PUB.*appaccount*	then
	SL.PUB.SYS	
PX	SL.PUB.*appaccount*	then
	SL.PUB.SYS	

where *logongroup* and *logonaccount* refer to the user's logon group and account and *appgroup* and *appaccount* refer to the location where the Network/S application resides, by default, NETWORKS and TELAMON respectively. Note that if Network/S has been placed in PUB.SYS, the only meaningful options for *Lib* are G, P and S since GX and PX would both refer to PUB.SYS.

When running the Native Mode version of Network/S on an MPE/iX based HP 3000, Network/S will attempt to load the specified procedure from the XL file identified by *lib*, which must be a valid filename.

See Appendix D, *Procedure Examples*, for an example **#RJSTAT** procedure.

For example, an **#RJSTAT** display to **$STDLIST** would appear as:

```
#RJSTAT
*******   RJE PERFORMANCE STATISTICS   *******
* ELAPSED REAL TIME:            00:00:06.389 *
* ELAPSED CPU TIME:             00:00:00.472 *
* RECORDS SENT TO HOST:(IN)     0            *
* CHARACTERS SENT TO HOST:      0            *
* RECORDS SENT FROM HOST:(OUT)  0            *
* CHARACTERS SENT FROM HOST:    0            *
* RJLINE COMMANDS:              0            *
* RJIN COMMANDS:                0            *
* RJOUT COMMANDS:               0            *
* RJLIST COMMANDS:              0            *
* RJPUNCH COMMANDS:             0            *
* RJIO COMMANDS:                0            *
* RJEOD COMMANDS:               0            *
* RJINFO COMMANDS:              0            *
* RJDEBUG COMMANDS:             0            *
* RJSTAT COMMANDS:              1            *
* RJCONTINUE COMMANDS:          0            *
* RJCMDFILE COMMANDS:           0            *
* RJHELP COMMANDS:              0            *
*************** Error Summary ***************
* TIMEOUTS:                     0            *
* ASYNC ERRORS:                 0            *
* ENQs received                 0            *
* NAKs received                 0            *
* TTDs received                 0            *
* WACKs received                0            *
* ENQs sent                     0            *
* NAKs sent                     0            *
* TTDs sent                     0            *
* WACKs sent                    0            *
* Retries                       0            *
*********************************************
```

RJSYS

Execute MPE commands

The **#RJSYS** command is identical to **#RJMPE** except that JCW and Variable substitution will be performed on the text of the **#RJSYS** command prior to its execution. JCWs and MPE/iX variables are identified by a "!" prefix and Variables are identified by a "%" prefix. For example:

```
#RJMPE SETVAR FILENAME 'MYFILE'
#RJCALC %R = 132
#RJCALC %B = 4
#RJSYS PURGE !FILENAME
#RJSYS FILE !FILENAME;REC=-%R,%B,F,ASCII
#RJOUT *!FILENAME
```

Syntax:

#**RJSYS** ***command***

CHAPTER 4

Other Topics

Defaults

Network/S provides a means to override the normal default values that apply when the program is first run. The program will search for a specially named initialization command file when initially run. This file can be placed in a variety of locations - Network/S will use the first one it finds. The specific locations for the file are:

```
RJECLINE.logongroup.logonacct   - then
RJECLINE.PUB.logonacct          - then
RJECLINE.PUB.SYS                - then
RJECLINE.NETWORKS.TELAMON
```

If one of the above files is found, it will be read start to finish.

Once an **#RJLINE** command has been read, Network/S will look for a device-specific initialization file. The name of the file is based on the actual device number of the port being accessed. If a device-class name was specified in the **#RJLINE** command, Network/S will first determine the device number. The search sequence is the same as that used for the **RJECLINE** file and the file's name is:

```
RJECL nnn
```

where *nnn* is the device number. If the device number is less than 100, the value will be zero-filled; if it is greater than 999, the value will overflow to the left.

The general entry format for the **RJECLINE/RJECL*nnn*** file is:

```
option {: | =} value
```

Leading, trailing and embedded blanks are ignored. Any unknown *option* is ignored as well. Valid commands include (default values, if any, are shown in UPPERCASE):

```
DEV=[logical device name/number] [,speed of DEV]
LINECODE={ASCII | ebcdic} [, {crc | LRC}]
ABIN={downloadable asynchronous firmware file name}
SBIN={downloadable synchronous firmware file name}
TABLE={file name}            ; As with RJLINE option
MODEM={DTR | uds | sadl | v.25[,ascii] | 801}
MODEMA={DTR | uds | sadl | v.25[,ascii]}
MODEMB={DTR | uds | sadl | v.25[,ascii]}
SELECT={MODEMA | modemb | either}
PREFIX="prefix string"
PRI=high[,abspri] | NORMAL
SHOW={YES | no}
VERBOSE={YES | no}
WARN={YES | no}
TRACE={yes | NO}
RETRY=n
CMD=RJE command
CONSOLE={yes | NO}
CRASH=n
SYN=n
NAK=n
ENQ=n
TTD=n
DIALWAIT=n
RTSCTS=n
```

Thus, to change the default **LINECODE** to **EBCDIC**, you would enter:

```
LINECODE=EBCDIC
```

in the **RJECLINE** file.

If you've attached your Synchronous Network Engine to logical device 101 and you don't want to reconfigure that port to have a device class name of RJLINE, you could enter:

```
DEV=101
```

If no **RJECLINE** file is found, Network/S proceeds with the normal defaults.

DEV

See **#RJLINE,DEV** for a detailed description of this option.

Syntax:

```
DEV=[logical device name/number] [,speed]
```

LINECODE

See **#RJLINE,LINECODE** for a detailed description of this option.

Syntax:

```
LINECODE={EBCDIC | ASCII} [ ,{CRC | LRC} ]
```

ABIN and SBIN

For use with downloadable Network Engines. These options specify the names of the text files containing the downloadable firmware code for the Network Engine. **ABIN** identifies the code for Asynchronous operation and **SBIN** identifies the code for Synchronous operation.

These files will be provided by Telamon.

Syntax:

```
ABIN=async-fw-filename
SBIN=sync-fw-filename
```

TABLE

See **#RJLINE,TABLE** for a detailed description of this option.

Syntax:

```
TABLE={file name}
```

MODEM

See **#RJLINE,CONNECT,DIAL** for a detailed description of this option.

Syntax:

```
MODEM={DTR[*] | SADL | UDS[*] | V.25[,ASCII] | 801}
```

MODEMA

For use with the Dual Modem Engine only.

This option identifies the modem auto-dialer type attached to the **MODEM A** port of the Network Engine. When the **SELECT=MODEMA** option is specified on the **#RJLINE** command, the modem type specified here will be used, by default.

Syntax:

```
MODEMA={DTR[*] | SADL | UDS[*] | V.25[,ASCII]}
```

MODEMB

For use with the Dual Modem Engine only.

This option identifies the modem auto-dialer type attached to the **MODEM B** port of the Network Engine. When the **SELECT=MODEMB** option is specified on the **#RJLINE** command, the modem type specified here will be used, by default.

Syntax:

```
MODEMB={DTR[*] | SADL | UDS[*] | V.25[,ASCII]}
```

SELECT

See **#RJLINE,SELECT** for a detailed description of this option.

Syntax:

```
SELECT={MODEMA | MODEMB | EITHER}
```

PREFIX

The **PREFIX** option allows you to specify the default dialing prefix for your telephone system.

For example, if you must dial "9" before all outgoing calls, you could specify:

```
PREFIX="9"
```

Syntax:

```
PREFIX="prefix string"
```

PRI

See **#RJLINE,PRI** for a detailed description of this option.

Syntax:

```
PRI={NORMAL | HIGH[,abspri]}
```

SHOW

See **#RJLINE,SHOW** for a detailed description of this option.

Syntax:

```
SHOW={YES | NO}
```

VERBOSE

See **#RJLINE,VERBOSE** for a detailed description of this option.

Syntax:

```
VERBOSE={YES | NO}
```

WARN

See **#RJLINE,WARN** for a detailed description of this option.

Syntax:

```
WARN={YES | NO}
```

TRACE

See **#RJLINE,TRACE** for a detailed description of this option.

Syntax:

```
TRACE={YES | NO}
```

RETRY

See **#RJLINE,RETRY** for a detailed description of this option.

Syntax:

```
RETRY=count
```

CMD

The **CMD** option provides the means to have Network/S execute a specific command when it is first run. If, for example, you've developed an **#RJCONTINUE** procedure to be used at all times, you could enter:

```
CMD=RJCONTINUE @MY'PROC(S)
```

to cause the procedure to be loaded whenever Network/S is run.

Syntax:

```
CMD=command
```

Default:

```
none
```

CONSOLE

The **CONSOLE** option will cause most error and warning messages to be displayed on the system console (via the PRINTOP intrinsic) as well as to **$STDLIST**.

Syntax:

```
CONSOLE={YES | NO}
```

Default:

```
CONSOLE=NO
```

CRASH

Specifies the crash timer. This timer is used to cause the Network Engine to reset itself in the event that the Network/S software and/or the local computer fails to communicate with it. If the designated time interval passes and the Engine receives no data or commands from the Network/S process, a reset will occur and any open connection will be terminated. Note that this timer is in effect while Network/S is awaiting commands at the "#" prompt - should the timer expire while Network/S is awaiting the next command, the Engine *will* reset and drop any connection.

The crash timer is specified in seconds, even though the value sent to the Engine is converted to an integral number of minutes. Consequently, the value specified should be a multiple of 60 seconds.

Specifying a value of zero (0) will disable the crash timer.

Syntax:

```
CRASH=n
```

Default:

```
CRASH=60      ; 60 seconds -> 1 minute
```

SYN

Specifies the number of **syn** (synchronous idle) characters that will be sent at the start of each packet of information sent by the Engine.

Syntax:

```
SYN=n
```

Default:

```
SYN=4
```

NAK

Specifies the maximum number of consecutive **nak**s (negative acknowledge) that can be *received* before a failure is reported. **nak**s are sent when checksum errors are detected in received data. In **3780** mode, **nak**s are also sent in response to received **ttd**s (see below) - these **nak**s are *not* counted with this parameter.

Syntax:

```
NAK=n
```

Default:

```
NAK=6
```

ENQ

Specifies the maximum number of consecutive **enq**s (enquire) that can be sent before failure. **enq**s are sent as a status check when the receiving system fails to respond to data sent.

Syntax:

```
ENQ=n
```

Default:

```
ENQ=6
```

TTD

Specifies the length of time (in 3-second intervals) between **ttd**s (temporary text delays.) **ttd**s are sent in **3780** mode when the Network Engine is in the process of sending a file and its Transmit buffer is empty. This usually occurs when the local computer system becomes overloaded and cannot send data to the Engine fast enough for the Engine to continue transmitting uninterrupted.

Syntax:

```
TTD=n
```

Default:

```
TTD=1
```

DIALWAIT

Specifies the maximum amount of time (in 4-second intervals) that the Engine will wait for a response to a dial sequence. Increase this value if your modem takes an unusually long time to connect.

Syntax:

```
DIALWAIT=n
```

Default:

```
DIALWAIT=20
```

RTSCTS

Specifies an RTS-CTS turnaround delay to be used in conjunction with that provided by the modem. If a non-zero value is specified, the Network Engine will delay between raising the RTS signal and waiting for CTS the indicated amount of time, expressed in milliseconds. This ensures that at least this much time will pass before any information is transmitted and can be used in conjunction with the modem's built-in RTS-CTS delay.

Syntax:

```
RTSCTS=n
```

Default:

```
RTSCTS=0
```

Job Control Words

Network/S sets a number of job control words (JCWs) during its execution.

NESERROR:

This JCW is set to zero (0) at program initiation, and is set to one (1) in the event of any error. It is the user's responsibility to reset this JCW to zero after an error has been detected. This is a general purpose flag that merely indicates that some other type of error has occurred - the other error types must be tested for, independently.

FSERROR:

This JCW is set to the MPE File System error number should some such error occur during program execution. For example, if an **#RJIN** command fails due to a non-existent permanent file, this JCW will be set to 52.

CSERROR:

This JCW is set to the second number shown in the message displayed after a communications error:

```
**** CS ERROR: a, b
```

message. For instance, when the line drops, this JCW will be set to 103. On a dialing error, it will be set to 54.

SUPPRESSCSALL:
SUPPRESSCS*nnn*:

These JCWs can be used to cause the suppression of CS error messages to the console. If **SUPPRESSCSALL** is set to 1, no CS error messages will be generated on the console. Use **SUPPRESSCS*nnn*** to specify individual error types to be suppressed. e.g.

```
:SETJCW SUPPRESSCS217 = 1
```

will prevent "**** CS ERROR: 2, 217" from being displayed on the console. The messages will be displayed upon $STDLIST regardless of the JCW settings.

ERRORCLASS:
ERRORTYPE:
ERRORNUM:

These JCWs are set to the values of the three-word array used by an **#RJCONTINUE** procedure. See the information on the **#RJCONTINUE** procedure in the *Procedure Examples* appendix for more information on these values.

NETWORK:

When and if Network/S terminates abnormally, this JCW is set to the specific internal error number that caused the termination.

Its significance is merely as a test as to whether Network/S terminated normally or not. If the value is zero (0), no error has occurred.

ROUTECODE:

For use with **#RJOUT** procedures. This JCW is set whenever a block is received that may contain a 3780 component select directive. The JCW is initialized to zero (0) and can take the following values:

```
1 - Route to List device
2 - Route to Punch device
3 - Unrouted
```

ETXRECEIVED:

For use with **#RJOUT** procedures. This JCW is set prior to calling the **#RJOUT** procedure with the last (or only) record from a block terminated with an ETX character. This is used to identify the last record in a data set, in order to provide the means to switch HP 3000 files between data-sets in a single **#RJOUT** reception. The JCW is set to one (1) when an ETX has been received, zero (0) otherwise.

EOTRECEIVED:

This JCW will be set whenever the most recent **#RJOUT** command has received an EOT code to terminate the file transfer. If set to zero (0) and the ETX keyword has been specified, the **#RJOUT** command has finished due to the receipt of an ETX-terminated block, not because of an EOT. If set to one (1), and EOT was received.

RJOUTCOUNT:

This JCW shows the number of files received during the execution of the most recent **#RJOUT** command. It is specifically intended for use when using the "filename,count" option, where the JCW will show how many files were actually received during the command's execution.

Utility Programs

Included with the Network/S software are a number of utility programs that may prove helpful to you.

DUMP.PUB.TELAMON

This application is used to display the contents of disc files using, by default, a combination of Hexadecimal and ASCII formats. To use it, type:

```
:RUN DUMP.PUB.TELAMON;INFO="filename"
```

The output will be displayed upon **$STDLIST**, one screen-full at a time. The cursor stops at the end of the last line that fits and **DUMP** will then await user input. Typing any character other than Carriage Return will terminate the program; typing a Carriage Return will cause the next screen-full of data to be displayed.

When you want to examine a file's contents exactly, using the **DUMP** program is much more reliable than attempting to view the file using your terminal's Display Functions or even by using Hewlett-Packard's FCOPY utility. DUMP displays *each* character found in each record in the file (Display Functions cannot display all control characters) - trailing blanks are *not* ignored and repeating sequences are *not* skipped (FCOPY attempts to shorten the display by eliminating what it considers redundant data.)

A number of options are available to modify the programs behavior. They include:

- -x ASCII dump in hex with no interpretation
- -a ASCII dump in hex with ASCII mnemonic interpretations (default)
- -e EBCDIC dump in hex with EBCDIC mnemonic interpretations. Use this if your file contains EBCDIC data. The printable text will be converted to ASCII and the mnemonics will be interpreted using the EBCDIC character set
- -o ASCII dump in Octal
- -r Interpret characters > $7F as Roman8. Normally characters with the high-order bit on are treated as non-printable characters
- -c Access file in 'COPY' mode, the same as specifying `COPY` in a `:FILE` equation for the file to be displayed
- -*n* Start dumping the file's contents at record #*n*. Note that records are numbered starting with zero (0)
- -w *n* Use 'n' as the $STDLIST record width
- -j *n* Use *n* as terminal screen height (default is 23)
- -p Send output to LP

These options are specified in the INFO string, prior to the name of the file to be dumped. For example, to dump a file called MYDATA, starting at record number 10, in normal Hex mode, enter:

```
:RUN DUMP.PUB.TELAMON:INFO="-10 -x MYDATA"
```

Run the **DUMP** program with no INFO string for a short display of the program's options.

CTLSCAN.PUB.TELAMON

This application is used to quickly examine data files for control characters. Very often, errors occur during the execution of the **#RJIN** command when it turns out that the file being transmitted contains certain control characters that can affect the transfer's outcome.

It is difficult, if not impossible, to detect these control characters, even when using a utility like the **DUMP** program, above. **CTLSCAN** can very quickly examine your file and provide a display and/or count of control characters found. To use it, type:

```
:RUN CTLSCAN.PUB.TELAMON;INFO="[-b] [-i] [-e] [-c] filename"
```

By default, **CTLSCAN** will display on your screen each record found that contains control characters. The offending characters will be displayed in inverse video on HP terminals.

A number of options are available to modify the programs behavior. They include:

- -b Shows only BSC control characters
- -i Shows questionable EBCDIC (IBM 3780 table) characters
- -e Combines default with -i option
- -c Counts selected characters without displaying each record
- -<n> Start test at record #n

These options are specified in the INFO string, prior to the name of the file to be dumped. For example, to examine a file called MYDATA, starting at record number 10, looking for a summary display, enter:

```
:RUN CTLSCAN.PUB.TELAMON:INFO="-10 -c MYDATA"
```

Run the **CTLSCAN** program with no INFO string for a short display of the program's options.

APPENDIX A Basic Concepts

At its simplest, most data communications can be described as voltage pulsing On and Off over a circuit between two end points. These On/Off signals, when examined at fixed time intervals, are interpreted as bits. Three circuits, or wires, are used to carry out a conversation between two end points: one circuit with which to Transmit bits, one circuit with which to Receive bits and a Ground circuit. In the RS-232 standard, *On*, or *1*, is identified by a minus twelve volt (-12V) state and *Off*, or *0*, by a plus twelve volt (+12V) state. Zero volts (0V) is neither *On* nor *Off*.

The Baud Rate specifies the fixed time interval, or clock, used to generate (send) and interpret (receive) bits on the circuit. This clock can either be generated internally by the sending device or can be provided externally, usually by a modem. The sender and receiver must use the same clock definition in order for individual bits to be recognized.

If, for example, the baud rate is 300 (sometimes referred to as 300 bits-per-second, or bps), the sender must hold the voltage for each bit On or Off, as the case may be, for 1/300th of a second. The receiver, on the other hand, samples the line every 1/300th of a second, determining at that point whether the line is *On* (-12V) or *Off* (+12V). (In fact, an asynchronous receiver samples at 16 times the baud rate, or every 1/4800th of a second. More on this later.)

The goal is to recognize bit patterns in this stream of incoming data. In most cases, data is interpreted in groups of eight-bit packets, or bytes. Applications then interpret these bytes as either individual character values, taken from the ASCII or EBCDIC character sets, or as numeric values ranging in value from 0 to 255 (larger numeric values are formed by combining two or more contiguous bytes).

Asynchronous Operation

As mentioned earlier, an asynchronous receiver samples the line at 16 times (16X) the baud rate. This is done to provide the means to handle the situation where the sender's clock signal may not be precisely in tune with the receiver's clock. The 16X sampling frequency allows for an up to 3% range of variability - differences in baud rates of 3% or less should still result in a recognizable byte.

In asynchronous operation, the sender can send a byte at any time, without any prior indication - e.g. asynchronously. As an indicator to the receiver, each byte is preceded by what is known as a Start Bit, with a value of *0*. In

the idle state, each transmitter sends a constant stream of *1* bits, referred to as *Marking*, at the configured baud rate. The receiver, sampling the line at 16X the baud rate, will observe the line switching from *1*'s to *0*'s (from -12V to +12V) between some two consecutive samples. To the receiver, this transition only marks the *beginning* of the Start Bit. The receiver then counts eight more sample times (half-way through the sample size) to position itself roughly at the middle of the incoming Start Bit. From that point on, the receiver examines the line state every 16 sample times, determining if the current bit is *On* or *Off* by measuring the voltage at the apparent middle of the bit. When the sender finishes sending the byte, it will revert to Marking for at least one bit's time. The receiver uses this last bit, or Stop Bit, as nominal verification that the byte was received correctly. If, instead, the receiver determines that the tenth bit (counting the Start Bit as the first bit) is a *0*, then an error has occurred and the byte is discarded - this is referred to as a Framing Error.

In some instances, the last of the eight data-bits is used as a validity check on the preceding seven bits. This type of error checking is known as Vertical Redundancy Checking (VRC), and the bit is referred to as the Parity bit. When specified, Parity can have one of the following definitions:

- Even - the Parity bit is set so that the sum of the parity bit and the remaining seven bits is an even value.
- Odd - the Parity bit is set so that the sum of the parity bit and the remaining seven bits is an odd value.
- Mark, or 1’s - the Parity bit is always set to one (1).
- Space, or 0’s - the Parity bit is always set to zero (0).

If the receiver's calculated Parity doesn't agree with actual Parity bit, then the byte is discarded and a Parity Error is generated.

Note: When Parity is utilized, only seven bits of significant data may be represented, with a numeric range of 0 to 127 ($00 to $7F.)

Since each byte must be preceded by the Start Bit and followed by at least one Stop Bit, a total of ten bits, at a minimum, must be sent for each byte. This is why we typically divide a baud rate value (9600) by ten to determine the maximum characters-per-second (CPS) rate (960) that can be obtained in asynchronous mode.

After having waited at least one Stop Bit's time, the sender can either send another byte if it has one to send, or it can continue Marking until another byte becomes available for transmission. Thus, the baud rate divided by ten indicates the *maximum* speed in characters-per-second - actual data rates are often significantly lower.

Note that for this scheme to work reliably, both the sender and receiver must have previously agreed upon the baud rate to be used. If different values are used, or if the sender's or receiver's clock frequency is too high or too low, they won't be able to communicate.

Note also that the two devices have to be connected in such a way as to guarantee that the first device's Transmit circuit is connected to the second device's Receive circuit, and vice-versa. At a minimum, a ground circuit must also be provided. This is important: if the wiring isn't correct, no communications can take place.

Synchronous Operation

With Synchronous communications, a different scheme is employed with respect to baud rates and character framing.

Rather than trusting that both sides are using the same internal clock speeds, clock information is shared. Three additional circuits are required, known as the Transmit Clock, the Receive Clock and the External Transmit Clock.

The External Transmit Clock is a signal generated by each device, indicating its own transmit clock speed. This clock's signal is sent to two destinations: the device's own Transmit Clock (an input signal) and the *other* device's Receive Clock (also an input signal.) Similarly, the other device's External Transmit Clock is shared between its own Transmit Clock and the originating device's Receive Clock. Thus, for each direction, the transmitter and receiver both utilize the same clock signal.

Note that with this arrangement, it is possible (albeit highly unlikely) that the two devices can exchange data at different baud rates.

Synchronous transmissions also adopt a scheme eliminating the Start and Stop bits utilized in Asynchronous transmissions, resulting in an immediate 20% savings in transmission time. Only eight bits are transmitted per byte, not ten. However, without these framing bits, a new scheme must be defined to indicate when the transmitter is ready to send data.

Two methods are currently in use. The first, and older, method is called Character (or Byte) Synchronous mode, where synchronization occurs after the recognition of one or more whole byte patterns. The other is called Bit Synchronous mode, where synchronization occurs after the recognition of bit patterns in sizes larger than eight bits. Bit Synchronous mode is utilized by the Systems Network Architecture (SNA) and Synchronous Data Link Control (SDLC) protocols and is beyond the scope of this discussion.

The data package used in asynchronous mode, consisting of one start bit, eight data bits and one stop bit, is replaced here with a multi-character package, consisting of a start sequence, a multi-character data block and a stop sequence. In this mode, groups of bytes are collected and sent in a contiguous block, as opposed to one-at-a-time, at (possibly) random intervals. The sender indicates the start of a block by sending a start sequence, consisting (in Byte Synchronous mode) of two or more pre-defined bytes, known as synchronization, or Sync, characters. The receiver, now sampling at the actual baud rate used by the transmitter (not 16 times the baud rate as mentioned earlier), collects eight bits at a time, in a register, examining each eight-bit pattern to see if it matches the Sync character. If the test fails, the receiver waits for the next bit to arrive, discarding the oldest bit in the register, adding the newest bit to the other end and then re-applying the test. This process is repeated until a successful match is made, at which point the receiver clears the register and awaits the next eight consecutive bits. If the second pattern fails to match the Sync character, the single bit scanning is resumed. On the other hand, if the second pattern also matches the Sync character, the receiver is said to be “in sync” with the transmitter. From this point on, until a pre-defined stop sequence is recognized, the receiver can take each subsequent eight-bit value and treat it as a byte.

Note: We refer to this mode of operation as bi-synchronous, the “bi” derived from the need to recognize at least two Sync characters at the start of each packet. Most bisync devices send more than two Sync characters (usually four) at the start of each packet to help improve the chances that the receiver will synchronize quickly. Once synchronized, the receiver will discard subsequent, consecutive, Sync characters.

Since only eight bits per byte are transmitted and received, Synchronous communications can achieve higher data rates since the baud rate can be divided by eight to determine the maximum CPS rate. Thus, 9600 baud yields a maximum data rate of 1,200 CPS in Synchronous mode as opposed to 960 CPS in Asynchronous mode,

an improvement of 25%. Due to the fact that Asynchronous characters can be transmitted with possibly substantial delays between characters, the throughput difference can be even greater.

Modems

While not explicitly stated in the RS-232 standard, the practical maximum cable length between two devices is only about 50 feet, or slightly more than 15 meters. To communicate over longer distances requires either the use of line drivers, capable of boosting the RS-232 voltages over longer distances, or modems, devices that convert the low-power digital On/Off signals described earlier to audible sounds that can be more easily transmitted over long distances, using telephone lines for example. The audible sounds are referred to as Analog signals.

When modems are used for Synchronous lines, the clock signals are not sent between the modems. Instead, the clock signals are generated and maintained by the modems for use between each modem and its attached device. As the modem typically provides both the Transmit and Receive Clock signals, the device's External Transmit Clock is usually not used.

Since the modems (also known as Data Communications Equipment or DCE devices) control the speed of their attached devices (also known as Data Terminal Equipment or DTE devices) the only real constraint is that the two modems be mutually compatible, observing the same Analog modulation protocol on the lines between the modems. Some common modem protocols and their corresponding baud rates are:

TABLE A-1.

Protocol	Speed
V.22	2400 baud
Bell 201	2400 baud
Bell 208	4800 baud
V.32	9600 baud

For reasons beyond the scope of this discussion, the aforementioned modems are mutually incompatible. Bell 201 modems can only communicate with other Bell 201 modems, Bell 208 modems with other Bell 208 modems, and so on. (V.32 modems can communicate with V.22 and other, slower modems, but not with Bell 201/208 modems.)

Duplex Modes

In our standard model, described earlier, it is possible for two DTE devices to be sending and receiving simultaneously. This is referred to as Full Duplex, sometimes as 4-wire mode. The major requirement is that the DTE devices be electronically able to send and receive data at the same time. The case where simultaneous send and receive is not possible is called Half Duplex, or 2-wire mode.

The older modem protocols, Bell 201 and Bell 208, operate in Half Duplex mode. As only one DTE may be transmitting at a time, some mechanism is required to allow the sending DTE to inform the receiving DTE that it wishes to send data. The mechanism employed involves the use of three additional circuits, two of which are used between the sending DTE and its DCE and one which is used between the receiving DCE and DTE devices.

The sending DTE indicates its desire to send by turning *On* a circuit, called Request-To-Send (RTS). The local DCE, in turn, sends an Analog carrier signal to the remote DCE (which then turns on the Data-Carrier-Detect (DCD) signal to its DTE) informing it that data is about to arrive, and after a configurable delay, turns On the

Clear-To-Send (CTS) signal back to the local DTE. This delay is sometimes referred to as the RTS-CTS Turn-around Delay. DTE devices configured for Half duplex operation are usually restricted in such a way that they cannot transmit data unless CTS is held high by the DCE and cannot attempt to receive data unless DCD is held high by the DCE. As we'll see later, the Network Engine is configured to operate in this fashion.

For example, the steps to send a message are:

1. DTE raises RTS and awaits CTS.
2. DCE (modem) sends analog carrier signal to remote DCE.
3. After the RTS-CTS Turnaround Delay, modem raises CTS back to its DTE.
4. DTE sends message.
5. DTE lowers RTS.
6. Modem drops analog carrier signal to remote modem and lowers CTS to its DTE.

When viewed from the modem's front panel, this sequence often takes place so quickly that it usually appears that the RTS, CTS and Transmit Data (TD) indicators turn On and Off simultaneously. (CTS may stay On with Full Duplex modems.)

From the receiving side, the sequence is:

1. When the receiving modem observes the analog carrier signal going On, it raises DCD to its DTE.
2. Message arrives.
3. When the modem observes the analog carrier signal going Off, it lowers DCD to its DTE.

When viewed from the modem's front panel, this sequence is seen by observing the DCD and Receive Data (RD) indicators turning On and Off, again nearly simultaneously. (DCD may stay On with Full Duplex modems.)

Newer modem protocols, such as V.22 and V.32, can operate in Full Duplex mode, where simultaneous send/receive is possible. The CTS and DCD signals are not used as handshaking signals in Full Duplex, although they are typically held high at all times.

The Synchronous Network Engine is designed to utilize the Half Duplex handshake, where transmission is inhibited unless CTS is present and DCD is required before any attempt is made to synchronize upon incoming data. Full Duplex modems may be used so long as they are configured to hold CTS and DCD high at all times.

RTS-CTS Turnaround

With Half Duplex modems, the RTS-CTS delay is a critical parameter. If the delay is set to too short a time, the sending modem may start sending its data packet before the receiving modem has enabled its receive logic. Remember that in bisync mode, the receiver must recognize at least two Sync characters before the in-bound packet can be received. If the RTS-CTS delay is too short, the Sync characters may very well be missed entirely - the rest of the packet will be ignored as the receiver never synchronized.

The need for a delay has to do with how the two modems communicate with one another and the transmission medium used. If the line quality isn't good or if there is a long delay introduced because of a satellite link, for instance, the RTS-CTS delay may need to be increased.

In general, the conservative approach is to select one of the higher values provided by the modem - most modems provide a simple choice of 50 milliseconds (ms), or five hundredths of a second, and 150ms. Some modems provide smaller values (5ms) and some larger (500ms.) The penalty for choosing a delay that is too short was mentioned earlier. The penalty for choosing a delay that is longer than necessary is reflected in decreased throughput. If, for example, a 50ms delay is all that is called for, but the modem is configured to provide a 150ms delay, an additional one tenth of a second delay will be added to *each* packet sent from the DTE. Send one hundred packets and add 10 seconds to the elapsed time; send one thousand packets and add 100 seconds; and so on.

Normally, it is the modem's responsibility to provide the RTS-CTS delay. The DTE is completely at the mercy of the modem. If the modem's maximum delay value proves to be too short, or if the modem doesn't seem to be waiting the correct amount of time, transmission becomes difficult, if not impossible. With Network Engine versions D.6.5 and later, it is now possible to effect a DTE-controlled RTS-CTS delay. The Engine will still raise RTS and wait for CTS, but will ensure that a minimum amount of time passes before attempting to transmit. If the Engine's delay is set to 500ms and the modem's delay is 150ms, CTS will be raised 150ms after the Engine raises RTS. The Engine will then wait another 350ms before transmitting.

Transfer Protocols

First, some simplifying assumptions.

- The mechanics of a 3780 file transfer will be described, although the fundamental concepts are applicable to 2780 file transfers as well as 3270 multi-point communications.
- A modem-to-modem connection will be implied, and for the scope of this explanation, we will assume that the modems are connected and on-line at all times.
- The examples will show communications using bisync control characters from the EBCDIC character set, although ASCII could, with very minor changes, be used as well.
- The RTS-CTS-DCD handshaking mechanism, while required in most cases, will be ignored here. For the sake of these examples, it will be assumed that each DTE/DCE pair will perform the appropriate RTS-CTS-DCD handshaking where called for.

While the RTS-CTS-DCD handshaking scheme, described earlier, indicates how packets of information are sent and received, additional rules must be utilized to control what exactly is being sent and what is expected to be received.

In the Idle, or Control, State, both of the DTE devices are Marking, as described earlier. When one of the DTE devices becomes ready to send data, it indicates its desire to do so by sending what is known as a Line Bid, which is simply a request to begin sending data. A simple Line Bid is in the form:

```
SYN SYN ENQ 'FF' 'FF'
```

where the **SYN** characters represent the Synchronization characters described earlier. See the **Glossary** section that follows for specific definitions. The **ENQ** character identifies this packet as a Line Bid. The **'FF'** characters are known as Pad characters, consisting of eight *1* bits. (**'FF'** identifies a byte with the Hex value of FF, or all *1* bits. It has no mnemonic representation and can only be shown in Hex form. When necessary, other similar Hex constants will be identified using the **'xx'** form.) The Pad characters are nothing more than what the receiver observes when the transmitter reverts to Marking, in this case after having sent the **ENQ** character. The receiving DTE interprets the Pad characters as the end of the incoming packet, although the end is usually recognized con-

textually - e.g. a Line Bid ends with the **ENQ**. Note that when using Half Duplex protocols, the DTE automatically drops out of synchronization when its modem drops the DCD signal. While they are typically seen (and sometimes required) on all packet types, Pads won't be shown for the rest of this discussion.

The other DTE, if it is ready and able to start receiving, indicates its readiness by sending a positive acknowledgment to the Line Bid. This takes the form:

```
SYN SYN DLE '70'
```

The **DLE '70'** is the EBCDIC code for an **ACK0**. We will see another type of acknowledgment, called **ACK1**, later.

If the receiving DTE isn't ready, it can either ignore the Line Bid, in which case the sender will usually repeat the Line Bid every two-to-three seconds, or it can explicitly reject the Bid by sending:

```
SYN SYN EOT
```

The **EOT** informs the sender that the receiver is not willing to accept data - the sender should then stop bidding.

If, on the other hand, the sender receives the **ACK0**, it will begin sending data. The sender is considered to now be in Transmit state and the receiver in Receive state. Data blocks take the general form:

```
SYN SYN STX|SOH data ETB|ETX ck
```

Data blocks can start with either an **STX**, or Start-of-Text, or (rarely) an **SOH**, or Start-of-Header. Data blocks are terminated with either an **ETB**, or End-of-Text-Block, or an **ETX**, or End-of-Text. The **ck** represents the block check code, either a one-byte Longitudinal Redundancy Check (LRC) or a two-byte Cyclical Redundancy Check (CRC) sequence. LRC codes are typically used with ASCII transmissions and CRC codes with EBCDIC transmissions.

The block check is used to verify that the data was received exactly as it was sent. For simplicity, we'll refer to this as the checksum, although neither the LRC nor CRC are checksums - they are calculated quite differently. The sender calculates the checksum starting with the first character following the **STX|SOH** and ending with the **ETB|ETX**, inclusive. The receiver performs the same calculations, starting its computation once the **STX|SOH** is recognized and stopping once the **ETB|ETX** has been seen. The next one, or two, byte field is then taken as the checksum for comparison with the calculated value. (The careful reader will observe that problems can arise if the **data** portion contains a byte that *looks*, to the receiver, like either an **ETB** or an **ETX** - more on this later.) If the checksums match, the receiver sends a positive acknowledge, in this case taking the form:

```
SYN SYN DLE /
```

The **DLE /** is the EBCDIC code for the **ACK1** mentioned earlier.

Note: After the initial **ACK0** response to the Line Bid, **ACK1** and **ACK0** codes are alternated on successive packets. This scheme is necessary to handle recovery in the event of lost data packets and/or lost acknowledgments. More on this later.

If the checksums do *not* match, the receiver indicates this with:

```
SYN SYN NAK
```

The **NAK**, or Negative-Acknowledge, indicates that the sender should re-transmit the packet just sent. The sender and receiver may both have limits as to the maximum number of such re-transmissions that may be

attempted. If the limit is exceeded, the transfer is stopped with the offended party sending an **EOT** sequence to indicate the fact. In general, the **EOT** sequence can be used unilaterally to cause both parties to revert to the Control state, which is its meaning here.

If, upon some subsequent re-transmission, the calculated checksum finally *does* match, the receiver sends the appropriate **ACK0/ACK1** and the sender then moves on to the next operation.

When the sender has finished sending, upon receipt of the final **ACK0/ACK1**, an **EOT** is sent to the receiver, placing both DTE devices back into the Control state. Note that there is no response to an **EOT** - the receiver "quietly" enters the Control state.

As an example, here is a short, successful, exchange:

Sender	Receiver	Explanation
`ENQ`		Line Bid
	`ACK0`	Bid Acknowledged
`STX data1 ETB ck`		#1 Received, checksum Ok
	`ACK1`	Positive Ack sent
`STX data2 ETB ck`		#2 Received, checksum Ok
	`ACK0`	Positive Ack sent
`STX data3 ETX ck`		#3 Received, checksum Ok
	`ACK1`	Positive Ack sent
`EOT`		End of Transmission

In this example, the first two data blocks have been shown with **ETB** terminators, and the last with an **ETX** terminator. This is the typical convention used in 2780/3780 emulation, although not a hard-and-fast rule. **ETB** is normally used to terminate all intermediate blocks, that is each but the last block, and **ETX** is used to terminate the final block in a transmission, the one immediately preceding the **EOT**. Again, this is a commonly accepted convention, not a rule.

By definition, a receiver can send a **NAK** packet in response to a data packet if, and only if, the data packet is "well formed", but with an invalid checksum. A well formed packet is one with a recognizable **STX|SOH** and a recognizable **ETB|ETX**. Only then is the checksum calculation attempted.

For example:

Sender	Receiver	Explanation
`ENQ`		Line Bid
	`ACK0`	Bid Acknowledged
`STX data1 ETB ck`		#1 Received, checksum Ok
	`ACK1`	Positive Ack sent
`STX corrupted data2 ETB ck`		Bad checksum
	`NAK`	Negative Acknowledge sent
`STX data2 ETB ck`		#2 Received, checksum Ok
	`ACK0`	Positive Ack sent
`STX data3 ETX ck`		#3 Received, checksum Ok
	`ACK1`	Positive Ack sent
`EOT`		End of Transmission

If the receiver doesn't see a well formed packet, the correct response is to do nothing - it is the sender's responsibility to attempt recovery. If, for example, the received packet's **SYN**'s aren't recognized because of an RTS-CTS delay problem at the sending end, the receiver will never have synchronized. The sender typically waits approximately two or three seconds for a reply from the receiver - if nothing is received after this time interval, the sender sends:

```
SYN SYN ENQ
```

which looks suspiciously like the Line Bid from above. In this context, however, the **ENQ** command is a directive to the receiver, telling it to re-transmit the last packet that *was* sent, either **ACK0**, **ACK1** or **NAK**, as the case may be. This scheme provides a completely reliable method for re-synchronization. If the data packet wasn't received, the **ACK** sent will be the **ACK** for the previous packet - the sender will realize this and re-transmit the data packet. On the other hand, if the data packet *was* received correctly, but the **ACK** was lost or garbled, the receiver will retransmit the current **ACK** and the sender will realize that the current block had been received and will move on to the next block. Similarly, if the data packet was received, but with an incorrect checksum, the receiver will retransmit the **NAK** packet.

For example, if the data block isn't received successfully, recovery looks like:

Sender	Receiver	Explanation
ENQ		Line Bid
	ACK0	Bid Acknowledged
STX data1 ETB ck		#1 Received, checksum Ok
	ACK1	Positive Ack sent
STX data2		Not “well formed”, no reply
		After three seconds...
ENQ		Send last Response
	ACK1	Previous Ack sent
STX data2 ETB ck		#2 Received, checksum Ok
	ACK0	Positive Ack sent
STX data3 ETX ck		#3 Received, checksum Ok
	ACK1	Positive Ack sent
EOT		End of Transmission

If the receiver does receive the packet, but the ACK is lost, recovery looks like:

Sender	Receiver	Explanation
ENQ		Line Bid
	ACK0	Bid Acknowledged
STX data1 ETB ck		#1 Received, checksum Ok
	ACK1	Positive Ack sent
STX data2 ETB ck		#2 Received, checksum Ok
	ACK0	Positive Ack sent, but lost
		Nothing after three seconds
ENQ		Send last Ack
	ACK0	Previous Ack sent
STX data3 ETX ck		#3 Received, checksum Ok
	ACK1	Positive Ack sent
EOT		End of Transmission

Command Language

To implement the Sender's side of the conversation, using the Synchronous Network software (Network/S), one could enter (commands and data are underlined):

```
#RJLINE 3780;LINECODE=EBCDIC;MAXRPB=1
#RJIN
#?data1
#?data2
#?data3
#?#RJEOD
#RJEND
```

Network/S will perform all of the necessary handshaking and packaging of the data. The user is responsible only for the content of the file or files to be transmitted.

If the three data records were instead in a file called INDATA, one could enter:

```
#RJLINE 3780;LINECODE=EBCDIC;MAXRPB=1
#RJIN INDATA
#RJEOD
#RJEND
```

The Receiver's side might be:

```
#RJLINE 3780;LINECODE=EBCDIC
#RJOUT
#RJEND
```

which would display the received data upon the terminal. To write the data to a file called OUTDATA instead:

```
#RJLINE 3780;LINECODE=EBCDIC
#RJOUT OUTDATA
#RJEND
```

2780/3780

IBM originally developed these protocols as a means to communicate with batch job entry terminals, consisting of a card reader, a line printer and, sometimes, a card punch. Users would submit jobs in the form of 80-column punched cards and receive their output on the line printer and/or card punch.

The IBM 2780 and 3780 Data Communication Terminals were the original devices used to communicate with the IBM mainframe computers using these protocols. Software products that perform these type of transfers are referred to as 2780 or 3780 Emulators. When files are transmitted, the card reader's function is emulated. When files are received, the line printer's and/or card punch's function is emulated.

The IBM 2780 is the predecessor of the 3780. There are a number of differences between the two devices, the most significant of which (for our purposes) is that the 2780's maximum transmission buffer size is 400 bytes and

the 3780's maximum is 512 bytes. The 3780 is also capable of performing what is known as Space Compression, a technique to improve transmission times by collapsing runs of space, or blank, characters into shorter sequences. For the most part, however, these two devices are very similar, and for the remainder of this discussion, the 3780 terminal's function will be described. It will be so noted when and where the 2780's function differs significantly.

As was noted earlier, every time a block of information is sent, the sender must stop and await the receiver's response before proceeding. This is called a *Line Turnaround*. If the RTS-CTS delay is in effect on either side, the Line Turnaround can be time consuming and can amount to a significant portion of the total elapsed time for the transmission. A number of techniques are available to minimize the number of Line Turnarounds.

Blocking

The most obvious option is to send more information at each turnaround. This is where the aforementioned maximum buffer sizes come into play. A 3780 terminal can send (and receive) up to 512 bytes at a time. Without any embellishments, this means that the terminal can send up to six whole card images, or records, (6 x 80 -> 480 bytes) at a time - neither 2780 nor 3780 terminals are capable of splitting records across buffers. Using this feature alone will reduce the number of line turnarounds by a factor of six - a significant reduction.

Truncation

Another option is to employ trailing blank *Truncation*. Many files to be sent contain information left-justified in the records with trailing blanks, or spaces. If the trailing blanks can be removed, it is often possible to send far more than six records in a block. The only problem is that a marker, or delimiter, now needs to be placed between the records in the buffer to mark where one shortened record stops and the next begins - it is no longer possible for the receiver to merely break the inbound buffer into fixed-length pieces. 3780 terminals use the Record Separator (**RS**) character for this purpose and 2780 terminals use the Unit Separator (**US**.) These characters are referred to as ITB characters. As records are read, trailing blanks are removed and the appropriate ITB is added before the data is written into the transmission buffer. Only when the buffer limit is reached or when some arbitrary number of records has been written will the buffer be transmitted. The receiver then *deblocks* the data using the ITB characters.

With blocking and truncation, our sample transmission might look like:

Sender	Receiver	Explanation
`ENQ`		Line Bid
	`ACK0`	Bid Acknowledged
`STX data1 RS data2 RS data3 RS ETX ck`		Received, checksum Ok
	`ACK1`	Positive Ack sent
`EOT`		End of Transmission

Compression

The last option is *Space Compression*. Very often, records contain contiguous runs of blank or space characters that can't be truncated, as they don't appear at the end of the record. 3780 terminals can reduce each run of blanks to a two-byte sequence consisting of the Group Separator (**GS**) character followed by a single byte identifying the number of blanks that have been compressed, from 3 to 63. (It is inefficient to compress any run of less than three bytes as it takes a minimum of two bytes to represent the sequence.) The scheme used to represent the length prevents values greater than 63 - if more than 63 blanks need to be compressed, consecutive **GS** pairs can be used. The receiver searches for the **GS** characters and inserts the indicated number of blanks back into the record. Obviously, the receiver must be capable of handling Space Compression for this technique to be used.

Transparency

The above techniques are usable so long as the data being sent contains mostly printable characters. If the data contains any of the aforementioned ITB (**RS/US**) or compression (**GS**) characters, or if it contains certain bisync control characters (**ENQ/ETB/ETX**), the data will have to be transmitted using what is known as *Transparent* mode. For example, if a record contains a byte that looks like an **ETB**, the receiver will mistakenly assume that this byte marks the end of a block and will attempt to compare its computed checksum with the next one or two bytes. As those bytes will most certainly *not* match the checksum, the receiver will **NAK** the packet repeatedly - the transmission will eventually fail.

In Transparent mode, data blocks take the form:

```
SYN SYN DLE {STX|SOH} data DLE {ETB|ETX} ck
```

The leading Data Link Escape (**DLE**) characters identify the block as a Transparent data block. Compression and Truncation are not permitted in transparent mode and records are placed in the buffer with no ITB characters. The receiver continues examining bytes until the two-byte end sequence is found, either **DLE ETB** or **DLE ETX**. The only catch is that the **DLE** character itself must be *guarded* if it appears in the data - otherwise, a two-byte data sequence containing a **DLE** followed by an **ETB** or **ETX** would cause the same problem as above. If **DLE** characters appear by themselves in the original data, the transmitter guards them by replacing each **DLE** with a **DLE DLE** pair. Thus, if the data contained:

```
A B C DLE D E ETX F
```

the transmission block would contain:

```
SYN SYN DLE STX A B C DLE DLE D E ETX F DLE ETB ck
```

The Receiver, switching to Transparent mode when the **DLE STX** is seen, converts **DLE DLE** pairs to single **DLE** characters and will only recognize **DLE ETB**, in this case, as the end of the block - the **ETX** will be interpreted, transparently, as data.

The trade-off is that while Transparent mode doesn't allow the efficiency features from above, it does provide the means to send *any* type of data - binary data files, program files, etc.

With transparency enabled, our sample transmission might look like:

Sender	Receiver	Explanation
ENQ		Line Bid
	ACK0	Bid Acknowledged
DLE STX data1 data2 data3 DLE ETX ck		Received, checksum Ok
	ACK1	Positive Ack sent
EOT		End of Transmission

3780 Operation

When a deck of cards is ready to be sent, the operator places the card deck into an input hopper on the 3780 terminal and presses the **START** switch. The reader then reads as many cards as can fit in the transmission buffer (after optional Truncation and/or Compression), adds an **ETB** code, calculates the checksum, transmits the buffer and awaits the appropriate acknowledgment. This is repeated until the hopper is emptied. If the card deck is too large to fit in the hopper, the operator places part of the deck into the hopper - if the hopper empties before the

operator re-fills it, the 3780 will stop and wait for the operator to add more cards and again press the START switch. (While the card reader is waiting for the operator, a special command sequence, a Temporary Text Delay (**TTD**), is sent to inform the receiver that the sender is temporarily out of data.) When the last (or only) portion of the deck has been placed into the hopper, the operator presses the **END OF FILE** switch before the START switch to enable the 3780 to terminate transmission once the hopper has emptied. This causes the final transmission block to be terminated with an **ETX** code and, upon receipt of an **ACK0/ACK1**, the 3780 sends an **EOT** and stops.

This behavior is mimicked by Network/S in the following fashion. Sending a file using the **#RJIN** command is analogous to the sending of a hopper full of cards. By default, when the file's end-of-file is encountered, Network/S withholds the last block of the file's data and waits for the next command before proceeding. (If too much time elapses before the next command, the Network Engine sends **TTD**s.) If the next command is another **#RJIN**, an **ETB** is added to the held block and it is sent - the next file's contents are then sent, the two files' contents appearing to the receiver to be one contiguous file. If, on the other hand, the next command is an **#RJOUT**, **#RJEOD** or **#RJEND**, an **ETX** is added to the held block, and, after it has been acknowledged, an **EOT** is sent. Network/S then proceeds with the indicated command.

The **#RJIN** command has keyword options controlling the Truncation option (**TRUNCATE**,) the Compression option (**COMPRESS**) and the Transparency option (**XPARENT**.) These options should be used *only* when it is known that the receiving system supports their use. Network/S automatically recognizes these features on incoming files - the **#RJOUT** command will expand and deblock the incoming data according to its content and format.

LINECODE, INCODE and OUTCODE

In most cases, 3780 emulators communicate using characters taken from the EBCDIC (Extended Binary Coded Decimal Interchange Code) character set. The other choice is USASCII (United States of America Standard Code for Information Interchange), or ASCII for short. This choice is identified by what is referred to here as **LINECODE**, or the character set to be used on the Communications Line. **LINECODE** is specified on the **#RJLINE** command.

Actual punched card data is encoded using the Hollerith coding scheme and the 3780 terminal translates these codes to EBCDIC or ASCII, as its configuration dictates. On the computer systems supporting Network/S, on the other hand, data is most often represented using the ASCII character set. Before this type of data can be sent, it needs to be translated to match the **LINECODE**. For the **#RJIN** command, the keyword **INCODE** informs Network/S as to the character set of the file being read - the character set cannot be determined by examining the file or its contents. Similarly, the **#RJOUT** command uses the keyword **OUTCODE** to indicate the character set to be used when writing the file. Both **INCODE** and **OUTCODE** default to ASCII.

When a file is sent, the data is translated from the **INCODE** to the **LINECODE** if the two values differ and if the **INCODE** value hasn't been set to **BINARY**, a special case to be used when sending data that is neither ASCII nor EBCDIC. Binary files are nearly always sent with the **XPARENT** option enabled as they usually contain bytes that may otherwise be interpreted as bisync control codes.

Similarly, when a file is being received, the data is translated from the **LINECODE** to the **OUTCODE** if the values differ and if **OUTCODE** isn't **BINARY**, as above.

Receiving Data

Host systems generally send files to 3780 emulators from their spooler sub-systems. If the host has multiple files to be sent, it can send them one at a time, each transmission starting with a Line Bid and ending with an **EOT**, or all of the files may be combined and sent as a single transmission, comprised of multiple *Data Sets*. This works quite well for print files sent to a printer as the files usually start on separate pages and are often preceded by header pages, allowing an operator to separate the data sets. Multiple data sets received via **#RJOUT**, on the other hand, can be difficult to distinguish from one another.

At the packet level, information is usually included that marks the boundary between data sets. The last block of a data set is terminated with the **ETX** character, as noted above. In this instance, however, it is not a requirement that the **ETX**-terminated block be followed by an **EOT** - another data block may follow, indicating the beginning of a new data set. By default, the **#RJOUT** command doesn't observe the **ETX** terminator as a file boundary, terminating instead only upon receipt of an **EOT**. By specifying the **ETX** keyword on the **#RJOUT** command, this behavior can be changed so that the **#RJOUT** function stops upon receipt of an **ETX**-terminated block, in much the same fashion as if an **EOT** had been received. See the Command Reference chapter for more information on this feature.

Print and Punch Files

Host systems can send Print and Punch data sets either as individual transmissions or multiple data sets can be combined into a single transmission. When combined, these file types are identified by the use of Component Select codes, or *Routing Codes*.

3780 routing codes, if used, appear in the first block of a data set, that is, either the first block following a Line Bid and/or the first block following an **ETX**-terminated block. The codes can take one of three values:

```
DC1    Route to Printer
DC2    Route to Punch1
DC3    Route to Punch2
```

Although only one card punch may be attached to a 3780, two codes are available for the punch - either value is acceptable. There are two variations of the **#RJOUT** command that are specifically designed to handle list and punch data: **#RJLIST** and **#RJPUNCH**. Their functions are virtually identical to those of the **#RJOUT** command with any differences noted below.

#RJLIST

The **#RJLIST** command will accept either routed Printer data sets or unrouted data sets. If a routed Punch data set is received, an error results. Printer data sets may also optionally contain Vertical Forms Control (VFC) or *Carriage Control* codes, specifying line spacing, page ejects, etc. 2780 and 3780 Carriage Control codes, when present, appear at the start of each record in a block, and are easily recognized as they consist of two-byte sequences, with the first byte being an Escape (**ESC**) character. If these codes are recognized, they are converted to the matching carriage control value for the system running Network/S before the data is written to the file.

#RJPUNCH

The **#RJPUNCH** command will accept either routed Punch data sets or unrouted data sets. If a routed Printer data set is received, an error results. On an HP3000, the default format of the punch file is slightly different than for List files - see the Command Reference chapter for more information.

#RJOUT

The **#RJOUT** command will accept routed Printer data sets, routed Punch data sets and/or unrouted data sets. It provides an additional feature in that the **#RJOUT** command can be specified so that routed Punch data sets are directed to a different file than routed Printer and unrouted data sets. In this mode, Network/S will automatically switch between two user specified files during the execution of the **#RJOUT** command, separating the Punch data from the Printer data. See the Command Reference chapter for more information on this feature.

2780 Routing

2780 routing codes, if used, precede each record in a block, in the form of Carriage Control, as with the Printer data sets above. Punch records are prefixed with a unique two-byte carriage control code; all other carriage control values are interpreted as Printer codes, causing the data to be treated as printer data. It is possible, although unlikely, for Print and Punch data to be combined in a single transmission block.

Glossary

RS-232 MODEM SIGNALS (SUBSET)

Mnemonic	Description	Pin	CCITT	EIA
FG	Frame Ground	1	101	AA
TD	Transmit Data	2	103	BA
RD	Receive Data	3	104	BB
RTS	Request To Send	4	105	CA
CTS	Clear To Send	5	106	CB
DSR	Data Set Ready	6	107	CC
SG	Signal Ground	7	102	AB
DCD	Data Carrier Detect	8	109	CF
TC	Transmit Clock	15	114	DB
RC	Receive Clock	17	115	DD
DTR	Data Terminal Ready	20	108.2	CD
RI	Ring Indicator	22	125	CE
(TC)	Ext. Transmit Clock	24	113	DA

BSC CONTROL CHARACTERS

Mnemonic	Description	EBCDIC	ASCII
SYN	Synchronous Idle	'32'	'16'
SOH	Start-of-Header	'01'	'01'
STX	Start-of-Text	'02'	'02'
ETX	End-of-Text	'03'	'03'
EOT	End-of-Transmission	'37'	'04'
ENQ	Enquire	'2D'	'05'
DLE	Data-Link-Escape	'10'	'10'
DC1	Device-Control-1	'11'	'11'
DC2	Device-Control-2	'12'	'12'
DC3	Device-Control-3	'13'	'13'
ESC	Escape	'27'	'1B'
NAK	Negative-Acknowledge	'3D'	'15'
ETB	End-of-Text-Block	'26'	'17'
GS	Group-Separator	'1D'	'1D'
RS	Record-Separator	'1E'	'1E'
US	Unit-Separator	'1F'	'1F'
TTD	Temporary Text Delay	STX ENQ	STX ENQ
ACK0	Acknowledge 0	DLE '70'	DLE 0
ACK1	Acknowledge 1	DLE /	DLE 1
WACK	Wait Before Transmit	DLE ,	DLE ;
RVI	Reverse-Interrupt	DLE @	DLE >
DLE-EOT	Switched Line Disconnect	DLE EOT	DLE EOT

APPENDIX B

Character Sets

Introduction

The following pages contain the ASCII-EBCDIC translate tables used by the Synchronous Network Engine. The first table is the default table, represented by the file **AEHP.NETWORKS.TELAMON**. The second table is an alternate table, represented by the file **AEIBM.NETWORKS.TELAMON**. The third table is an alternate table, represented by the file **AEATT.NETWORKS.TELAMON**. See the **#RJLINE** and **RJECLINE TABLE** parameter for more information on using these tables.

ASCII/EBCDIC (HP) Character Set

Char Code			Graphic		To EBCDIC		To ASCII	
Dec	Oct	Hex	ASC	EBC	Oct	Hex	Oct	Hex
0	000	00	NUL	NUL	000	00	000	00
1	001	01	SOH	SOH	001	01	001	01
2	002	02	STX	STX	002	02	002	02
3	003	03	ETX	ETX	003	03	003	03
4	004	04	EOT	PF	067	37	234	9C
5	005	05	ENQ	HT	055	2D	011	09
6	006	06	ACK	LC	056	2E	206	86
7	007	07	BEL	DEL	057	2F	177	7F
8	010	08	BS		026	16	227	97
9	011	09	HT	RLF	005	05	215	8D
10	012	0A	LF	SMM	045	25	216	8E
11	013	0B	VT	VT	013	0B	013	0B
12	014	0C	FF	FF	014	0C	014	0C
13	015	0D	CR	CR	015	0D	015	0D
14	016	0E	SO	SO	016	0E	016	0E
15	017	0F	SI	SI	017	0F	017	0F
16	020	10	DLE	DLE	020	10	020	10
17	021	11	DC1	DC1	021	11	021	11
18	022	12	DC2	DC2	022	12	022	12
19	023	13	DC3	DC3	023	13	023	13
20	024	14	DC4	RES	074	3C	235	9D
21	025	15	NAK	NL	075	3D	205	85
22	026	16	SYN	BS	062	32	010	08
23	027	17	ETB	IL	046	26	207	87
24	030	18	CAN	CAN	030	18	030	18
25	031	19	EM	EM	031	19	031	19
26	032	1A	SUB	CC	077	3F	222	92
27	033	1B	ESC	CU1	047	27	217	8F
28	034	1C	FS	IFS	034	1C	034	1C
29	035	1D	GS	IGS	035	1D	035	1D
30	036	1E	RS	IRS	036	1E	036	1E
31	037	1F	US	IUS	037	1F	037	1F
32	040	20	SP	DS	100	40	200	80
33	041	21	!	SOS	117	4F	201	81
34	042	22	"		177	7F	202	82
35	043	23	#		173	7B	203	83
36	044	24	$	BYP	133	5B	204	84
37	045	25	%	LF	154	6C	012	0A
38	046	26	&	ETB	120	50	027	17
39	047	27	'	ESC	175	7D	033	1B
40	050	28	(		115	4D	210	88
41	051	29	)		135	5D	211	89
42	052	2A	*	SM	134	5C	212	8A
43	053	2B	+	CU2	116	4E	213	8B
44	054	2C	,		153	6B	214	8C
45	055	2D	-	ENQ	140	60	005	05
46	056	2E	.	ACK	113	4B	006	06
47	057	2F	/	BEL	141	61	007	07
48	060	30	0		360	F0	220	90
49	061	31	1		361	F1	221	91
50	062	32	2	SYN	362	F2	026	16
51	063	33	3		363	F3	223	93
52	064	34	4	PN	364	F4	224	94
53	065	35	5		365	F5	225	95
54	066	36	6	UC	366	F6	226	96
55	067	37	7	EOT	367	F7	004	04
56	070	38	8		370	F8	230	98
57	071	39	9		371	F9	231	99
58	072	3A	:		172	7A	232	9A
59	073	3B	;	CU3	136	5E	233	9B
60	074	3C	<	DC4	114	4C	024	14
61	075	3D	=	NAK	176	7E	025	15
62	076	3E	>		156	6E	236	9E
63	077	3F	?	SUB	157	6F	032	1A

ASCII/EBCDIC (HP) Character Set (Continued)

Char Code			Graphic		To EBCDIC		To ASCII	
Dec	Oct	Hex	ASC	EBC	Oct	Hex	Oct	Hex
64	100	40	@	SP	174	7C	040	20
65	101	41	A		301	C1	240	A0
66	102	42	B		302	C2	241	A1
67	103	43	C		303	C3	242	A2
68	104	44	D		304	C4	243	A3
69	105	45	E		305	C5	244	A4
70	106	46	F		306	C6	245	A5
71	107	47	G		307	C7	246	A6
72	110	48	H		310	C8	247	A7
73	111	49	I		311	C9	250	A8
74	112	4A	J	[	321	D1	133	5B
75	113	4B	K	.	322	D2	056	2E
76	114	4C	L	<	323	D3	074	3C
77	115	4D	M	(	324	D4	050	28
78	116	4E	N	+	325	D5	053	2B
79	117	4F	O	!	326	D6	041	21
80	120	50	P	&	327	D7	046	26
81	121	51	Q		330	D8	251	A9
82	122	52	R		331	D9	252	AA
83	123	53	S		342	E2	253	AB
84	124	54	T		343	E3	254	AC
85	125	55	U		344	E4	255	AD
86	126	56	V		345	E5	256	AE
87	127	57	W		346	E6	257	AF
88	130	58	X		347	E7	260	B0
89	131	59	Y		350	E8	261	B1
90	132	5A	Z	]	351	E9	135	5D
91	133	5B	[	$	112	4A	044	24
92	134	5C	\	*	340	E0	052	2A
93	135	5D	]	)	132	5A	051	29
94	136	5E	^	;	137	5F	073	3B
95	137	5F	_	^	155	6D	136	5E
96	140	60	`	-	171	79	055	2D
97	141	61	a	/	201	81	057	2F
98	142	62	b		202	82	262	B2
99	143	63	c		203	83	263	B3
100	144	64	d		204	84	264	B4
101	145	65	e		205	85	265	B5
102	146	66	f		206	86	266	B6
103	147	67	g		207	87	267	B7
104	150	68	h		210	88	270	B8
105	151	69	i		211	89	271	B9
106	152	6A	j	|	221	91	174	7C
107	153	6B	k	,	222	92	054	2C
108	154	6C	l	%	223	93	045	25
109	155	6D	m	_	224	94	137	5F
110	156	6E	n	>	225	95	076	3E
111	157	6F	o	?	226	96	077	3F
112	160	70	p		227	97	272	BA
113	161	71	q	F1	230	98	273	BB
114	162	72	r	F2	231	99	274	BC
115	163	73	s	F3	242	A2	275	BD
116	164	74	t	F4	243	A3	276	BE
117	165	75	u	F5	244	A4	277	BF
118	166	76	v	F6	245	A5	300	C0
119	167	77	w	F7	246	A6	301	C1
120	170	78	x	F8	247	A7	302	C2
121	171	79	y	`	250	A8	140	60
122	172	7A	z	:	251	A9	072	3A
123	173	7B	{	#	300	C0	043	23
124	174	7C	|	@	152	6A	100	40
125	175	7D	}	'	320	D0	047	27
126	176	7E	~	=	241	A1	075	3D
127	177	7F	DEL	"	007	07	042	22

ASCII/EBCDIC (HP) Character Set (Continued)

Char Code			Graphic		To EBCDIC		To ASCII	
Dec	Oct	Hex	ASC	EBC	Oct	Hex	Oct	Hex
128	200	80			040	20	303	C3
129	201	81		a	041	21	141	61
130	202	82		b	042	22	142	62
131	203	83		c	043	23	143	63
132	204	84		d	044	24	144	64
133	205	85		e	025	15	145	65
134	206	86		f	006	06	146	66
135	207	87		g	027	17	147	67
136	210	88		h	050	28	150	68
137	211	89		i	051	29	151	69
138	212	8A			052	2A	304	C4
139	213	8B			053	2B	305	C5
140	214	8C			054	2C	306	C6
141	215	8D			011	09	307	C7
142	216	8E			012	0A	310	C8
143	217	8F			033	1B	311	C9
144	220	90			060	30	312	CA
145	221	91		j	061	31	152	6A
146	222	92		k	032	1A	153	6B
147	223	93		l	063	33	154	6C
148	224	94		m	064	34	155	6D
149	225	95		n	065	35	156	6E
150	226	96		o	066	36	157	6F
151	227	97		p	010	08	160	70
152	230	98		q	070	38	161	71
153	231	99		r	071	39	162	72
154	232	9A			072	3A	313	CB
155	233	9B			073	3B	314	CC
156	234	9C			004	04	315	CD
157	235	9D			024	14	316	CE
158	236	9E			076	3E	317	CF
159	237	9F			341	E1	320	D0
160	240	A0			101	41	321	D1
161	241	A1		~	102	42	176	7E
162	242	A2		s	103	43	163	73
163	243	A3		t	104	44	164	74
164	244	A4		u	105	45	165	75
165	245	A5		v	106	46	166	76
166	246	A6		w	107	47	167	77
167	247	A7		x	110	48	170	78
168	250	A8		y	111	49	171	79
169	251	A9		z	121	51	172	7A
170	252	AA			122	52	322	D2
171	253	AB			123	53	323	D3
172	254	AC			124	54	324	D4
173	255	AD			125	55	325	D5
174	256	AE			126	56	326	D6
175	257	AF			127	57	327	D7
176	260	B0			130	58	330	D8
177	261	B1			131	59	331	D9
178	262	B2			142	62	332	DA
179	263	B3			143	63	333	DB
180	264	B4			144	64	334	DC
181	265	B5			145	65	335	DD
182	266	B6			146	66	336	DE
183	267	B7			147	67	337	DF
184	270	B8			150	68	340	E0
185	271	B9			151	69	341	E1
186	272	BA			160	70	342	E2
187	273	BB			161	71	343	E3
188	274	BC			162	72	344	E4
189	275	BD			163	73	345	E5
190	276	BE			164	74	346	E6
191	277	BF			165	75	347	E7

ASCII/EBCDIC (HP) Character Set (Continued)

Char Code			Graphic		To EBCDIC		To ASCII	
Dec	Oct	Hex	ASC	EBC	Oct	Hex	Oct	Hex
192	300	C0		{	166	76	173	7B
193	301	C1		A	167	77	101	41
194	302	C2		B	170	78	102	42
195	303	C3		C	200	80	103	43
196	304	C4		D	212	8A	104	44
197	305	C5		E	213	8B	105	45
198	306	C6		F	214	8C	106	46
199	307	C7		G	215	8D	107	47
200	310	C8		H	216	8E	110	48
201	311	C9		I	217	8F	111	49
202	312	CA			220	90	350	E8
203	313	CB			232	9A	351	E9
204	314	CC			233	9B	352	EA
205	315	CD			234	9C	353	EB
206	316	CE			235	9D	354	EC
207	317	CF			236	9E	355	ED
208	320	D0		}	237	9F	175	7D
209	321	D1		J	240	A0	112	4A
210	322	D2		K	252	AA	113	4B
211	323	D3		L	253	AB	114	4C
212	324	D4		M	254	AC	115	4D
213	325	D5		N	255	AD	116	4E
214	326	D6		O	256	AE	117	4F
215	327	D7		P	257	AF	120	50
216	330	D8		Q	260	B0	121	51
217	331	D9		R	261	B1	122	52
218	332	DA			262	B2	356	EE
219	333	DB			263	B3	357	EF
220	334	DC			264	B4	360	F0
221	335	DD			265	B5	361	F1
222	336	DE			266	B6	362	F2
223	337	DF			267	B7	363	F3
224	340	E0		\	270	B8	134	5C
225	341	E1			271	B9	237	9F
226	342	E2		S	272	BA	123	53
227	343	E3		T	273	BB	124	54
228	344	E4		U	274	BC	125	55
229	345	E5		V	275	BD	126	56
230	346	E6		W	276	BE	127	57
231	347	E7		X	277	BF	130	58
232	350	E8		Y	312	CA	131	59
233	351	E9		Z	313	CB	132	5A
234	352	EA			314	CC	364	F4
235	353	EB			315	CD	365	F5
236	354	EC			316	CE	366	F6
237	355	ED			317	CF	367	F7
238	356	EE			332	DA	370	F8
239	357	EF			333	DB	371	F9
240	360	F0		0	334	DC	060	30
241	361	F1		1	335	DD	061	31
242	362	F2		2	336	DE	062	32
243	363	F3		3	337	DF	063	33
244	364	F4		4	352	EA	064	34
245	365	F5		5	353	EB	065	35
246	366	F6		6	354	EC	066	36
247	367	F7		7	355	ED	067	37
248	370	F8		8	356	EE	070	38
249	371	F9		9	357	EF	071	39
250	372	FA			372	FA	372	FA
251	373	FB			373	FB	373	FB
252	374	FC			374	FC	374	FC
253	375	FD			375	FD	375	FD
254	376	FE			376	FE	376	FE
255	377	FF		EO	377	FF	377	FF

ASCII/EBCDIC (IBM 3780) Character Set

Char Code			Graphic		To EBCDIC		To ASCII	
Dec	Oct	Hex	ASC	EBC	Oct	Hex	Oct	Hex
0	000	00	NUL	NUL	000	00	000	00
1	001	01	SOH	SOH	001	01	001	01
2	002	02	STX	STX	002	02	002	02
3	003	03	ETX	ETX	003	03	003	03
4	004	04	EOT	PF	067	37	234	9C
5	005	05	ENQ	HT	055	2D	011	09
6	006	06	ACK	LC	056	2E	206	86
7	007	07	BEL	DEL	057	2F	177	7F
8	010	08	BS		026	16	227	97
9	011	09	HT	RLF	005	05	215	8D
10	012	0A	LF	SMM	045	25	216	8E
11	013	0B	VT	VT	013	0B	013	0B
12	014	0C	FF	FF	014	0C	014	0C
13	015	0D	CR	CR	015	0D	015	0D
14	016	0E	SO	SO	016	0E	016	0E
15	017	0F	SI	SI	017	0F	017	0F
16	020	10	DLE	DLE	020	10	020	10
17	021	11	DC1	DC1	021	11	021	11
18	022	12	DC2	DC2	022	12	022	12
19	023	13	DC3	DC3	023	13	023	13
20	024	14	DC4	RES	074	3C	235	9D
21	025	15	NAK	NL	075	3D	205	85
22	026	16	SYN	BS	062	32	010	08
23	027	17	ETB	IL	046	26	207	87
24	030	18	CAN	CAN	030	18	030	18
25	031	19	EM	EM	031	19	031	19
26	032	1A	SUB	CC	077	3F	222	92
27	033	1B	ESC	CU1	047	27	217	8F
28	034	1C	FS	IFS	034	1C	034	1C
29	035	1D	GS	IGS	035	1D	035	1D
30	036	1E	RS	IRS	036	1E	036	1E
31	037	1F	US	IUS	037	1F	037	1F
32	040	20	SP	DS	100	40	200	80
33	041	21	!	SOS	132	5A	201	81
34	042	22	"		177	7F	202	82
35	043	23	#		173	7B	203	83
36	044	24	$	BYP	133	5B	204	84
37	045	25	%	LF	154	6C	012	0A
38	046	26	&	ETB	120	50	027	17
39	047	27	'	ESC	175	7D	033	1B
40	050	28	(		115	4D	210	88
41	051	29	)		135	5D	211	89
42	052	2A	*	SM	134	5C	212	8A
43	053	2B	+	CU2	116	4E	213	8B
44	054	2C	,		153	6B	214	8C
45	055	2D	-	ENQ	140	60	005	05
46	056	2E	.	ACK	113	4B	006	06
47	057	2F	/	BEL	141	61	007	07
48	060	30	0		360	F0	220	90
49	061	31	1		361	F1	221	91
50	062	32	2	SYN	362	F2	026	16
51	063	33	3		363	F3	223	93
52	064	34	4	PN	364	F4	224	94
53	065	35	5		365	F5	225	95
54	066	36	6	UC	366	F6	226	96
55	067	37	7	EOT	367	F7	004	04
56	070	38	8		370	F8	230	98
57	071	39	9		371	F9	231	99
58	072	3A	:		172	7A	232	9A
59	073	3B	;	CU3	136	5E	233	9B
60	074	3C	<	DC4	114	4C	024	14
61	075	3D	=	NAK	176	7E	025	15
62	076	3E	>		156	6E	236	9E
63	077	3F	?	SUB	157	6F	032	1A

ASCII/EBCDIC (IBM 3780) Character Set (Continued)

Char Code			Graphic		To EBCDIC		To ASCII	
Dec	Oct	Hex	ASC	EBC	Oct	Hex	Oct	Hex
64	100	40	@	SP	174	7C	040	20
65	101	41	A		301	C1	240	A0
66	102	42	B		302	C2	300	C0
67	103	43	C		303	C3	314	CC
68	104	44	D		304	C4	310	C8
69	105	45	E		305	C5	304	C4
70	106	46	F		306	C6	342	E2
71	107	47	G		307	C7	324	D4
72	110	48	H		310	C8	265	B5
73	111	49	I		311	C9	267	B7
74	112	4A	J		321	D1	277	BF
75	113	4B	K	.	322	D2	056	2E
76	114	4C	L	<	323	D3	074	3C
77	115	4D	M	(	324	D4	050	28
78	116	4E	N	+	325	D5	053	2B
79	117	4F	O		326	D6	251	A9
80	120	50	P	&	327	D7	046	26
81	121	51	Q		330	D8	305	C5
82	122	52	R		331	D9	301	C1
83	123	53	S		342	E2	315	CD
84	124	54	T		343	E3	311	C9
85	125	55	U		344	E4	325	D5
86	126	56	V		345	E5	321	D1
87	127	57	W		346	E6	335	DD
88	130	58	X		347	E7	331	D9
89	131	59	Y		350	E8	336	DE
90	132	5A	Z	!	351	E9	041	21
91	133	5B	[	$	260	B0	044	24
92	134	5C	\	*	340	E0	052	2A
93	135	5D	]	)	273	BB	051	29
94	136	5E	^	;	272	BA	073	3B
95	137	5F	_	¬	155	6D	252	AA
96	140	60	`	-	171	79	055	2D
97	141	61	a	/	201	81	057	2F
98	142	62	b		202	82	242	A2
99	143	63	c		203	83	330	D8
100	144	64	d		204	84	241	A1
101	145	65	e		205	85	340	E0
102	146	66	f		206	86	341	E1
103	147	67	g		207	87	320	D0
104	150	68	h		210	88	264	B4
105	151	69	i		211	89	266	B6
106	152	6A	j	|	221	91	174	7C
107	153	6B	k	,	222	92	054	2C
108	154	6C	l	%	223	93	045	25
109	155	6D	m	_	224	94	137	5F
110	156	6E	n	>	225	95	076	3E
111	157	6F	o	?	226	96	077	3F
112	160	70	p		227	97	326	D6
113	161	71	q	F1	230	98	334	DC
114	162	72	r	F2	231	99	244	A4
115	163	73	s	F3	242	A2	245	A5
116	164	74	t	F4	243	A3	243	A3
117	165	75	u	F5	244	A4	345	E5
118	166	76	v	F6	245	A5	246	A6
119	167	77	w	F7	246	A6	247	A7
120	170	78	x	F8	247	A7	346	E6
121	171	79	y	`	250	A8	140	60
122	172	7A	z	:	251	A9	072	3A
123	173	7B	{	#	300	C0	043	23
124	174	7C	|	@	152	6A	100	40
125	175	7D	}	'	320	D0	047	27
126	176	7E	~	=	241	A1	075	3D
127	177	7F	DEL	"	007	07	042	22

ASCII/EBCDIC (IBM 3780) Character Set (Continued)

Char Code			Graphic		To EBCDIC		To ASCII	
Dec	Oct	Hex	ASC	EBC	Oct	Hex	Oct	Hex
128	200	80			040	20	322	D2
129	201	81		a	041	21	141	61
130	202	82		b	042	22	142	62
131	203	83		c	043	23	143	63
132	204	84		d	044	24	144	64
133	205	85		e	025	15	145	65
134	206	86		f	006	06	146	66
135	207	87		g	027	17	147	67
136	210	88		h	050	28	150	68
137	211	89		i	051	29	151	69
138	212	8A			052	2A	373	FB
139	213	8B			053	2B	375	FD
140	214	8C			054	2C	344	E4
141	215	8D			011	09	354	EC
142	216	8E			012	0A	361	F1
143	217	8F			033	1B	376	FE
144	220	90			060	30	263	B3
145	221	91		j	061	31	152	6A
146	222	92		k	032	1A	153	6B
147	223	93		l	063	33	154	6C
148	224	94		m	064	34	155	6D
149	225	95		n	065	35	156	6E
150	226	96		o	066	36	157	6F
151	227	97		p	010	08	160	70
152	230	98		q	070	38	161	71
153	231	99		r	071	39	162	72
154	232	9A			072	3A	371	F9
155	233	9B			073	3B	372	FA
156	234	9C			004	04	327	D7
157	235	9D			024	14	362	F2
158	236	9E			076	3E	323	D3
159	237	9F			341	E1	272	BA
160	240	A0			101	41	261	B1
161	241	A1		~	144	64	176	7E
162	242	A2		s	142	62	163	73
163	243	A3		t	164	74	164	74
164	244	A4		u	162	72	165	75
165	245	A5		v	163	73	166	76
166	246	A6		w	166	76	167	77
167	247	A7		x	167	77	170	78
168	250	A8		y	276	BE	171	79
169	251	A9		z	117	4F	172	7A
170	252	AA			137	5F	270	B8
171	253	AB			275	BD	271	B9
172	254	AC			352	EA	343	E3
173	255	AD			375	FD	353	EB
174	256	AE			373	FB	360	F0
175	257	AF			372	FA	364	F4
176	260	B0		[	274	BC	133	5B
177	261	B1			240	A0	273	BB
178	262	B2			277	BF	274	BC
179	263	B3			220	90	374	FC
180	264	B4			150	68	276	BE
181	265	B5			110	48	275	BD
182	266	B6			151	69	363	F3
183	267	B7			111	49	367	F7
184	270	B8			252	AA	370	F8
185	271	B9			253	AB	365	F5
186	272	BA		^	237	9F	136	5E
187	273	BB		]	261	B1	135	5D
188	274	BC			262	B2	260	B0
189	275	BD			265	B5	253	AB
190	276	BE			264	B4	250	A8
191	277	BF			112	4A	262	B2

ASCII/EBCDIC (IBM 3780) Character Set (Continued)

Char Code			Graphic		To EBCDIC		To ASCII	
Dec	Oct	Hex	ASC	EBC	Oct	Hex	Oct	Hex
192	300	C0		{	102	42	173	7B
193	301	C1		A	122	52	101	41
194	302	C2		B	313	CB	102	42
195	303	C3		C	333	DB	103	43
196	304	C4		D	105	45	104	44
197	305	C5		E	121	51	105	45
198	306	C6		F	316	CE	106	46
199	307	C7		G	336	DE	107	47
200	310	C8		H	104	44	110	48
201	311	C9		I	124	54	111	49
202	312	CA			315	CD	366	F6
203	313	CB			335	DD	302	C2
204	314	CC			103	43	316	CE
205	315	CD			123	53	312	CA
206	316	CE			314	CC	306	C6
207	317	CF			334	DC	352	EA
208	320	D0		}	147	67	175	7D
209	321	D1		J	126	56	112	4A
210	322	D2		K	200	80	113	4B
211	323	D3		L	236	9E	114	4C
212	324	D4		M	107	47	115	4D
213	325	D5		N	125	55	116	4E
214	326	D6		O	160	70	117	4F
215	327	D7		P	234	9C	120	50
216	330	D8		Q	143	63	121	51
217	331	D9		R	130	58	122	52
218	332	DA			354	EC	356	EE
219	333	DB			374	FC	303	C3
220	334	DC			161	71	317	CF
221	335	DD			127	57	313	CB
222	336	DE			131	59	307	C7
223	337	DF			353	EB	357	EF
224	340	E0		\	145	65	134	5C
225	341	E1			146	66	237	9F
226	342	E2		S	106	46	123	53
227	343	E3		T	254	AC	124	54
228	344	E4		U	214	8C	125	55
229	345	E5		V	165	75	126	56
230	346	E6		W	170	78	127	57
231	347	E7		X	356	EE	130	58
232	350	E8		Y	355	ED	131	59
233	351	E9		Z	357	EF	132	5A
234	352	EA			317	CF	254	AC
235	353	EB			255	AD	337	DF
236	354	EC			215	8D	332	DA
237	355	ED			376	FE	350	E8
238	356	EE			332	DA	347	E7
239	357	EF			337	DF	351	E9
240	360	F0		0	256	AE	060	30
241	361	F1		1	216	8E	061	31
242	362	F2		2	235	9D	062	32
243	363	F3		3	266	B6	063	33
244	364	F4		4	257	AF	064	34
245	365	F5		5	271	B9	065	35
246	366	F6		6	312	CA	066	36
247	367	F7		7	267	B7	067	37
248	370	F8		8	270	B8	070	38
249	371	F9		9	232	9A	071	39
250	372	FA			233	9B	257	AF
251	373	FB			212	8A	256	AE
252	374	FC			263	B3	333	DB
253	375	FD			213	8B	255	AD
254	376	FE			217	8F	355	ED
255	377	FF		EO	377	FF	377	FF

ASCII/EBCDIC (ATT) Character Set

Char Code			Graphic		To EBCDIC		To ASCII	
Dec	Oct	Hex	ASC	EBC	Oct	Hex	Oct	Hex
0	000	00	NUL	NUL	000	00	000	00
1	001	01	SOH	SOH	001	01	001	01
2	002	02	STX	STX	002	02	002	02
3	003	03	ETX	ETX	003	03	003	03
4	004	04	EOT	PF	067	37	234	9C
5	005	05	ENQ	HT	055	2D	011	09
6	006	06	ACK	LC	056	2E	206	86
7	007	07	BEL	DEL	057	2F	177	7F
8	010	08	BS		026	16	227	97
9	011	09	HT	RLF	005	05	215	8D
10	012	0A	LF	SMM	045	25	216	8E
11	013	0B	VT	VT	013	0B	013	0B
12	014	0C	FF	FF	014	0C	014	0C
13	015	0D	CR	CR	015	0D	015	0D
14	016	0E	SO	SO	016	0E	016	0E
15	017	0F	SI	SI	017	0F	017	0F
16	020	10	DLE	DLE	020	10	020	10
17	021	11	DC1	DC1	021	11	021	11
18	022	12	DC2	DC2	022	12	022	12
19	023	13	DC3	DC3	023	13	023	13
20	024	14	DC4	RES	074	3C	235	9D
21	025	15	NAK	NL	075	3D	205	85
22	026	16	SYN	BS	062	32	010	08
23	027	17	ETB	IL	046	26	207	87
24	030	18	CAN	CAN	030	18	030	18
25	031	19	EM	EM	031	19	031	19
26	032	1A	SUB	CC	077	3F	222	92
27	033	1B	ESC	CU1	047	27	217	8F
28	034	1C	FS	IFS	034	1C	034	1C
29	035	1D	GS	IGS	035	1D	035	1D
30	036	1E	RS	IRS	036	1E	036	1E
31	037	1F	US	IUS	037	1F	037	1F
32	040	20	SP	DS	100	40	200	80
33	041	21	!	SOS	132	5A	201	81
34	042	22	"		177	7F	202	82
35	043	23	#		173	7B	203	83
36	044	24	$	BYP	133	5B	204	84
37	045	25	%	LF	112	4A	012	0A
38	046	26	&	ETB	120	50	027	17
39	047	27	'	ESC	175	7D	033	1B
40	050	28	(		115	4D	210	88
41	051	29	)		135	5D	211	89
42	052	2A	*	SM	134	5C	212	8A
43	053	2B	+	CU2	116	4E	213	8B
44	054	2C	,		153	6B	214	8C
45	055	2D	-	ENQ	140	60	005	05
46	056	2E	.	ACK	113	4B	006	06
47	057	2F	/	BEL	141	61	007	07
48	060	30	0		360	F0	220	90
49	061	31	1		361	F1	221	91
50	062	32	2	SYN	362	F2	026	16
51	063	33	3		363	F3	223	93
52	064	34	4	PN	364	F4	224	94
53	065	35	5		365	F5	225	95
54	066	36	6	UC	366	F6	226	96
55	067	37	7	EOT	367	F7	004	04
56	070	38	8		370	F8	230	98
57	071	39	9		371	F9	231	99
58	072	3A	:		172	7A	232	9A
59	073	3B	;	CU3	136	5E	233	9B
60	074	3C	<	DC4	114	4C	024	14
61	075	3D	=	NAK	176	7E	025	15
62	076	3E	>		156	6E	236	9E
63	077	3F	?	SUB	157	6F	032	1A

ASCII/EBCDIC (ATT) Character Set (Continued)

Char Code			Graphic		To EBCDIC		To ASCII	
Dec	Oct	Hex	ASC	EBC	Oct	Hex	Oct	Hex
64	100	40	@	SP	174	7C	040	20
65	101	41	A		301	C1	240	A0
66	102	42	B		302	C2	300	C0
67	103	43	C		303	C3	314	CC
68	104	44	D		304	C4	310	C8
69	105	45	E		305	C5	304	C4
70	106	46	F		306	C6	342	E2
71	107	47	G		307	C7	324	D4
72	110	48	H		310	C8	265	B5
73	111	49	I		311	C9	267	B7
74	112	4A	J	%	321	D1	045	25
75	113	4B	K	.	322	D2	056	2E
76	114	4C	L	<	323	D3	074	3C
77	115	4D	M	(	324	D4	050	28
78	116	4E	N	+	325	D5	053	2B
79	117	4F	O	|	326	D6	174	7C
80	120	50	P	&	327	D7	046	26
81	121	51	Q		330	D8	305	C5
82	122	52	R		331	D9	301	C1
83	123	53	S		342	E2	315	CD
84	124	54	T		343	E3	311	C9
85	125	55	U		344	E4	325	D5
86	126	56	V		345	E5	321	D1
87	127	57	W		346	E6	335	DD
88	130	58	X		347	E7	331	D9
89	131	59	Y		350	E8	336	DE
90	132	5A	Z	!	351	E9	041	21
91	133	5B	[	$	255	AD	044	24
92	134	5C	\	*	340	E0	052	2A
93	135	5D	]	)	275	BD	051	29
94	136	5E	^	;	137	5F	073	3B
95	137	5F	_	^	155	6D	136	5E
96	140	60	`	-	171	79	055	2D
97	141	61	a	/	201	81	057	2F
98	142	62	b		202	82	242	A2
99	143	63	c		203	83	330	D8
100	144	64	d		204	84	241	A1
101	145	65	e		205	85	340	E0
102	146	66	f		206	86	341	E1
103	147	67	g		207	87	320	D0
104	150	68	h		210	88	264	B4
105	151	69	i		211	89	266	B6
106	152	6A	j		221	91	251	A9
107	153	6B	k	,	222	92	054	2C
108	154	6C	l		223	93	277	BF
109	155	6D	m	_	224	94	137	5F
110	156	6E	n	>	225	95	076	3E
111	157	6F	o	?	226	96	077	3F
112	160	70	p		227	97	326	D6
113	161	71	q	F1	230	98	334	DC
114	162	72	r	F2	231	99	244	A4
115	163	73	s	F3	242	A2	245	A5
116	164	74	t	F4	243	A3	243	A3
117	165	75	u	F5	244	A4	345	E5
118	166	76	v	F6	245	A5	246	A6
119	167	77	w	F7	246	A6	247	A7
120	170	78	x	F8	247	A7	346	E6
121	171	79	y	`	250	A8	140	60
122	172	7A	z	:	251	A9	072	3A
123	173	7B	{	#	300	C0	043	23
124	174	7C	|	@	117	4F	100	40
125	175	7D	}	'	320	D0	047	27
126	176	7E	~	=	241	A1	075	3D
127	177	7F	DEL	"	007	07	042	22

ASCII/EBCDIC (ATT) Character Set (Continued)

Char Code			Graphic		To EBCDIC		To ASCII	
Dec	Oct	Hex	ASC	EBC	Oct	Hex	Oct	Hex
128	200	80			040	20	322	D2
129	201	81		a	041	21	141	61
130	202	82		b	042	22	142	62
131	203	83		c	043	23	143	63
132	204	84		d	044	24	144	64
133	205	85		e	025	15	145	65
134	206	86		f	006	06	146	66
135	207	87		g	027	17	147	67
136	210	88		h	050	28	150	68
137	211	89		i	051	29	151	69
138	212	8A			052	2A	373	FB
139	213	8B			053	2B	375	FD
140	214	8C			054	2C	344	E4
141	215	8D			011	09	354	EC
142	216	8E			012	0A	361	F1
143	217	8F			033	1B	376	FE
144	220	90			060	30	263	B3
145	221	91		j	061	31	152	6A
146	222	92		k	032	1A	153	6B
147	223	93		l	063	33	154	6C
148	224	94		m	064	34	155	6D
149	225	95		n	065	35	156	6E
150	226	96		o	066	36	157	6F
151	227	97		p	010	08	160	70
152	230	98		q	070	38	161	71
153	231	99		r	071	39	162	72
154	232	9A			072	3A	371	F9
155	233	9B			073	3B	372	FA
156	234	9C			004	04	327	D7
157	235	9D			024	14	362	F2
158	236	9E			076	3E	323	D3
159	237	9F			341	E1	272	BA
160	240	A0			101	41	261	B1
161	241	A1		~	144	64	176	7E
162	242	A2		s	142	62	163	73
163	243	A3		t	164	74	164	74
164	244	A4		u	162	72	165	75
165	245	A5		v	163	73	166	76
166	246	A6		w	166	76	167	77
167	247	A7		x	167	77	170	78
168	250	A8		y	276	BE	171	79
169	251	A9		z	152	6A	172	7A
170	252	AA			272	BA	270	B8
171	253	AB			273	BB	271	B9
172	254	AC			352	EA	343	E3
173	255	AD		[	375	FD	133	5B
174	256	AE			373	FB	360	F0
175	257	AF			372	FA	364	F4
176	260	B0			274	BC	353	EB
177	261	B1			240	A0	273	BB
178	262	B2			277	BF	274	BC
179	263	B3			220	90	374	FC
180	264	B4			150	68	276	BE
181	265	B5			110	48	275	BD
182	266	B6			151	69	363	F3
183	267	B7			111	49	367	F7
184	270	B8			252	AA	370	F8
185	271	B9			253	AB	365	F5
186	272	BA			237	9F	252	AA
187	273	BB			261	B1	253	AB
188	274	BC			262	B2	260	B0
189	275	BD		]	265	B5	135	5D
190	276	BE			264	B4	250	A8
191	277	BF			154	6C	262	B2

ASCII/EBCDIC (ATT) Character Set (Continued)

Char Code			Graphic		To EBCDIC		To ASCII	
Dec	Oct	Hex	ASC	EBC	Oct	Hex	Oct	Hex
192	300	C0		{	102	42	173	7B
193	301	C1		A	122	52	101	41
194	302	C2		B	313	CB	102	42
195	303	C3		C	333	DB	103	43
196	304	C4		D	105	45	104	44
197	305	C5		E	121	51	105	45
198	306	C6		F	316	CE	106	46
199	307	C7		G	336	DE	107	47
200	310	C8		H	104	44	110	48
201	311	C9		I	124	54	111	49
202	312	CA			315	CD	366	F6
203	313	CB			335	DD	302	C2
204	314	CC			103	43	316	CE
205	315	CD			123	53	312	CA
206	316	CE			314	CC	306	C6
207	317	CF			334	DC	352	EA
208	320	D0		}	147	67	175	7D
209	321	D1		J	126	56	112	4A
210	322	D2		K	200	80	113	4B
211	323	D3		L	236	9E	114	4C
212	324	D4		M	107	47	115	4D
213	325	D5		N	125	55	116	4E
214	326	D6		O	160	70	117	4F
215	327	D7		P	234	9C	120	50
216	330	D8		Q	143	63	121	51
217	331	D9		R	130	58	122	52
218	332	DA			354	EC	356	EE
219	333	DB			374	FC	303	C3
220	334	DC			161	71	317	CF
221	335	DD			127	57	313	CB
222	336	DE			131	59	307	C7
223	337	DF			353	EB	357	EF
224	340	E0		\	145	65	134	5C
225	341	E1			146	66	237	9F
226	342	E2		S	106	46	123	53
227	343	E3		T	254	AC	124	54
228	344	E4		U	214	8C	125	55
229	345	E5		V	165	75	126	56
230	346	E6		W	170	78	127	57
231	347	E7		X	356	EE	130	58
232	350	E8		Y	355	ED	131	59
233	351	E9		Z	357	EF	132	5A
234	352	EA			317	CF	254	AC
235	353	EB			260	B0	337	DF
236	354	EC			215	8D	332	DA
237	355	ED			376	FE	350	E8
238	356	EE			332	DA	347	E7
239	357	EF			337	DF	351	E9
240	360	F0		0	256	AE	060	30
241	361	F1		1	216	8E	061	31
242	362	F2		2	235	9D	062	32
243	363	F3		3	266	B6	063	33
244	364	F4		4	257	AF	064	34
245	365	F5		5	271	B9	065	35
246	366	F6		6	312	CA	066	36
247	367	F7		7	267	B7	067	37
248	370	F8		8	270	B8	070	38
249	371	F9		9	232	9A	071	39
250	372	FA			233	9B	257	AF
251	373	FB			212	8A	256	AE
252	374	FC			263	B3	333	DB
253	375	FD			213	8B	255	AD
254	376	FE			217	8F	355	ED
255	377	FF		EO	377	FF	377	FF

APPENDIX C

Modem Configurations

Racal/Vadic 4850PA and 9650PA

If you've attached a Racal/Vadic or Racal/Milgo modem to the Network Engine, first make sure that the modem is correctly configured before starting. Set up the modem as follows:

Options listed as "x" may be ignored; options listed as "?" are site and/or user dependent - recommended settings are marked with "*"; all other options must be set as specified.

```
01*2 - Standard Options             Default Disabled
02*3 - DSR Control                  DSR normal
03*x - Reserved
04*? - Auto-answer                  1-Enabled*
                                    2-Disabled
05*? - Dial Mode                    1-Auto
                                    2-Tone*
                                    3-Pulse
06*? - Blind Dial                   1-Disabled (K::=wait for tone)
                                    2-Enabled  (K::=5 seconds)
07*x - Reserved
08*? - Answer Delay                 1-First ring*
                                    2-Third ring
09*x - Reserved
10*? - Failed Call Timer            1-60 Seconds*
                                    2-30 Seconds
                                    3-Disabled
11*x - Reserved
12*x - Reserved
13*? - Speed Select                 1-2400 bps
                                    2-unused
                                    3-4800 bps*
14*? - Busy Out with DTR LOW        1-Disabled*
                                    2-Enabled
15*1 - NRZI                         1-Enabled*
                                    2-Disabled
16*? - Satellite Option             1-Disabled*
                                    2-Enabled
17*? - Clear To Send Delay          1-50ms*
                                    2-150ms
18*1 - Local Copy                   Disabled
19*? - Turnaround Delay             1-Disabled*
                                    2-Enabled
20*1 - Line Current Disconnect      1-Enabled
21*1 - Clock                        1-Internal
22*1 - Force DTR                    1-Disabled
23*x - ACR Timing                   Don't care
24*x - Issue ACR on Invalid Digit   Don't care
25*x - Issue ACR on Busy Tone Detect Don't care
26*x - Issue ACR on Local Modem Busy Don't care
27*1 - Interactive Character Set    1-EBCDIC
28*1 - Modulation                   1-208B
29*1 - Receiver Equalization       1-Disabled
```

Load these options from the control panel on the modem as per the modem's instructions.

This modem is dialed using the **SADL** option. This option may be specified with the **CONNECT=DIAL** sequence or the **MODEM** keyword in the **RJECLINE** file.

UDS V.3225/V.3225L

If you've attached the UDS V.3225 or V.3225L modem to the Network Engine, first make sure that the modem is correctly configured before starting.

The key-stroke sequence to configure this modem follows. The keys on the front of the modem are labeled:

```
YES  NO  TALK/DATA
```

The UDS V.3225 and the UDS V.3225L have nine different factory default settings. Configuring these modems consists of selecting one of the factory defaults and changing three individual parameters. All custom settings are lost when a factory default setting is selected!

The configuration sequence is:

Menu (as it appears on the LCD display)	Key
V.32 9600 IDLE	NO
DIAL STORED NUMBER?	NO
DISPLAY STATUS?	NO
SELECT TEST?	NO
MODIFY CONFIGURATION?	YES
CHANGE MODEM OPTIONS?	NO
CHANGE MNP OPTIONS?	NO
CHANGE DTE OPTIONS?	NO
CHANGE TEST OPTIONS?	NO
CHANGE DIALER LINE OPTIONS?	NO
CHANGE SPEAKER OPTIONS?	NO
LOAD OR STORE OPTION SET?	YES
LOAD FACTORY OPTION SET?	YES
ARE YOU SURE?	YES
LOAD FACTORY OPTION SET #1?	NO
LOAD FACTORY OPTION SET #2?	NO
LOAD FACTORY OPTION SET #3?	YES
FACTORY OPTION SET #3 LOADED.	*modem pauses for a few seconds*
STORE PRESENT OPTIONS?	NO
LOAD OR STORE OPTION SET?	NO
MODIFY CONFIGURATION?	YES
CHANGE MODEM OPTIONS?	NO
CHANGE MNP OPTIONS?	NO
CHANGE DTE OPTIONS?	YES
SYNC DATA CHANGE TO ASYNC?	NO
MANUAL DIALER. CHANGE?	YES
CHANGE DIALER TO V.25 BISYNC?	YES
CHANGE V.25 BISYNC DIALER?	NO
CHARACTER TYPE ASCII CHANGE?	YES
CHARACTER TYPE EBCDIC CHANGE?	NO
DTR DISCONNECTS (&D2) CHANGE?	NO
DSR IS NORMAL CHANGE?	NO
DCD IS NORMAL CHANGE?	YES
CHANGE TO DCD FORCED HIGH?	YES
DCD FORCED HIGH CHANGE?	NO

```
CTS FOLLOWS RTS CHANGE?                 NO
RTS-CTS DELAY IS 0ms. CHANGE?           NO
DTE FALLBACK DISABLED.  CHANGE?         TALK/DATA
CHANGE DTE OPTIONS?                     TALK/DATA
LOAD OR STORE OPTION SET?               YES
LOAD FACTORY OPTION SET?                NO
STORE PRESENT OPTIONS?                  YES
ARE YOU SURE?                           YES
PRESENT OPTIONS ARE NOW PERMANENT.      modem pauses for a few seconds
                                        TALK/DATA

V.32 9600 IDLE
```

This modem is dialed using the **V.25** option. This option may be specified with the **CONNECT=DIAL** sequence or the **MODEM** keyword in the **RJECLINE** file.

UDS V.3229/V.3229L

If you've attached the UDS V.3229 or V.3229L modem to the Network Engine, first make sure that the modem is correctly configured before starting.

The key-stroke sequence to configure this modem follows. The keys on the front of the modem are labeled:

```
YES  NO  TALK/DATA
```

The UDS V.3229 and the UDS V.3229L have nine different factory default settings. Configuring these modems consists of selecting one of the factory defaults and changing three individual parameters. All custom settings are lost when a factory default setting is selected!

The configuration sequence is:

Menu (as it appears on the LCD display)	Key
V.32b 14400 IDLE	NO
DIAL STORED NUMBER?	NO
DISPLAY STATUS?	NO
SELECT TEST?	NO
MODIFY CONFIGURATION?	YES
CHANGE MODEM PARAMETERS?	NO
CHANGE PROTOCOL PARAMETERS?	NO
CHANGE DTE PARAMETERS?	NO
CHANGE TEST PARAMETERS?	NO
CHANGE DIAL LINE PARAMETERS?	NO
CHANGE SPEAKER OPERATION?	NO
LOAD OR STORE OPTION SET?	YES
LOAD FACTORY OPTION SET?	YES
LOAD FACTORY OPTION SET #1?	NO
LOAD FACTORY OPTION SET #2?	NO
LOAD FACTORY OPTION SET #3?	NO
LOAD FACTORY OPTION SET #4?	NO
LOAD FACTORY OPTION SET #5?	NO
LOAD FACTORY OPTION SET #6?	NO
LOAD FACTORY OPTION SET #7?	NO
LOAD FACTORY OPTION SET #8?	NO
LOAD FACTORY OPTION SET #9?	YES
FACTORY OPTION SET #9 LOADED.	*modem pauses for a few seconds*
STORE PRESENT OPTIONS?	NO
USER OPTION 1 AT RESET. CHANGE?	NO
LOAD OR STORE OPTION SET?	NO
MODIFY CONFIGURATION?	YES
CHANGE MODEM PARAMETERS?	NO
CHANGE PROTOCOL OPTIONS?	NO
CHANGE DTE OPTIONS?	YES
SYNC DATA CHANGE TO ASYNC?	NO
V.25 SDLC DIALER. CHANGE?	YES
CHANGE DIALER TO ASYNC COMMANDED?	NO
CHANGE DIALER TO DTR CONTROLLED?	NO
CHANGE DIALER TO MANUAL?	NO

```
CHANGE DIALER TO V.25 BISYNC?          YES
V.25 BISYNC DIALER.  CHANGE?           NO
CHARACTER TYPE ASCII CHANGE?           YES
CHARACTER TYPE EBCDIC CHANGE?          NO
DTR DISCONNECTS (&D2) CHANGE?          NO
DSR IS NORMAL  CHANGE?                 NO
DCD IS NORMAL  CHANGE?                 YES
CHANGE TO DCD FORCED HIGH?             YES
DCD FORCED HIGH  CHANGE?               NO
CTS FOLLOWS RTS CHANGE?                NO
RTS-CTS DELAY IS 0ms. CHANGE?          NO
DTE FALLBACK DISABLED.  CHANGE?        TALK/DATA
CHANGE DTE PARAMETERS?                 TALK/DATA
LOAD OR STORE OPTION SET?              YES
LOAD FACTORY OPTION SET?               NO
LOAD FROM USER OPTION SET?             NO
STORE PRESENT OPTIONS?                 YES
STORE TO USER OPTION SET #1?           YES
USER OPTION SET #1 STORED.             modem pauses for a few seconds
USER OPTION 1 AT RESET. CHANGE?        NO
LOAD OR STORE OPTION SET?              TALK/DATA
MODIFY CONFIGURATION?                  TALK/DATA
V.32b 14400 IDLE
```

This modem is dialed using the **V.25** option. This option may be specified with the **CONNECT=DIAL** sequence or the **MODEM** keyword in the **RJECLINE** file.

Other UDS Modems

If you've attached either of the UDS 201C/D, 208B/D, 9600B/D or 2140 modems or one of the Black Box model 201C/D or 208B/D modems to the Network Engine, the following discussion describes the recommended configuration settings. Since the various modems have different switch and jumper settings, we will list the configuration settings in generic terms - you will have to consult the manual that came with your modem for more specific instructions on how to implement the configuration.

Note that some of the following settings will be site and condition dependent. For instance, the CTS Delay and Carrier Detect Level settings may need to be adjusted according to the response characteristics of either the remote system or the telephone network used.

- Output Level Select: set to -9dBm
- Line Reversal: Normal
- DTE LLB: "D" position (Disabled)
- Receive Equalizer: set as is appropriate for your line quality
- Chassis Ground to Signal Ground: Strap for Signal Ground
- CTS Delay: set as is appropriate for your communications needs
- INT/EXT Transmitter clock: set for internal clocking
- Line Current Disconnect: set for 90ms
- Receiver and Call Turnaround Squelch: set as is appropriate
- Auto or Manual Answer: set for Auto answer if you expect incoming calls
- Carrier Detect Level: set for -30dBm for good phone lines
- DSR in Dial Mode: set to Normal
- No Activity Abort: set to Normal
- Satellite Options: set as is appropriate

Codex 2264

If you've attached a Codex 2264 modem to the Network Engine, first make sure that the modem is correctly configured before starting.

Set up the modem by first choosing Option Set 3, as explained in the modem's manual.

Then, change the following items:

Under "Modify Terminal", change:

```
Setting:          From:     To:
DCD:              Normal    ACU On
```

Under "Modify V.25 bis", change:

```
SyncProtocol:     Bit       Char
Char Set:         ASCII     EBCDIC
ACU Idle:         Char      Mark
```

This modem is dialed using the **V.25** option. This option may be specified with the **CONNECT=DIAL** sequence or the **MODEM** keyword in the **RJECLINE** file.

Codex 3260

If you've attached a Codex 3260 modem to the Network Engine, first make sure that the modem is correctly configured before starting.

Reset the modem to factory defaults by selecting the "Reinit Memory" option. Then, set up the modem by choosing "Select Options=3", as explained in the modem's manual.

Then, change the following items (under the indicated menus):

Under "ACU OPT'S", change:

```
Setting          From      To
V25 Form:        Bitsync   Charsync
V25 Char:        ASCII     EBCDIC
V25 Idle:        Char      Mark
```

Under "TERMINAL OPT'S", change:

```
DCD:             Normal    Acu On
```

Finally, enter "Save Changes=3" to save the configuration in the modem's non-volatile memory.

This modem is dialed using the **V.25** option. This option may be specified with the **CONNECT=DIAL** sequence or the **MODEM** keyword in the **RJECLINE** file.

USRobotics Courier V.32

Note that this modem can only dial one phone number, the number stored in the modem's non-volitile RAM. Changing phone numbers is difficult because you must reset the modem's DIP switches to factory defaults, connect an ASCII terminal to change the phone number, then reset the DIP switches back to bisynchronous operation.

If you've attached a USRobotics Courier HST Dual Standard or V.32 modem to the Network Engine, first make sure that the modem is correctly configured before starting. Set up the modem as follows:

```
AT&F                    Reset to factory defaults
ATB0|1                  0=V.32 mode|1=HST mode
AT&X0                   Internal Transmit Clock
AT&M1                   Synchronous Mode
AT&R0                   Delay before CTS after RTS
ATS0=0|1|...|n          0 Disables auto-answer, ring count otherwise
ATS13=0|8               8 auto-dials from &Z0 register
ATS26=n  {5}            RTS-CTS delay = 10*n ms
AT&Z0=phone number      Used if S13=8, above
AT&S1                   Normal DSR
AT&C1                   Modem controls CD
AT&D2                   DTE Controls DTR
AT&W                    Save configuration in NRAM
```

These options must be loaded interactively, using an ASCII terminal connected to the DTE port on the modem. Set the terminal for 9600 baud, 8 data bits and no parity for this operation.

The modem's DIP switches should be set as follows:

```
QUAD     1  2  3  4  5  6  7  8  9  10
 U       U  U  D  U  ?  U  D  D  D  U    (U=Off or Open; D=On or Closed)
 |       |  |  |  |  |  |  |  |  |  |--> Load from NVRAM on power-up
 |       |  |  |  |  |  |  |  |  |-----> Don't hangup on +++
 |       |  |  |  |  |  |  |  |--------> Smart mode
 |       |  |  |  |  |  |  |-----------> No result codes in Answer mode
 |       |  |  |  |  |  |--------------> Normal DCD on pin 8
 |       |  |  |  |  |-----------------> Auto answer (U=Yes, D=No)
 |       |  |  |  |--------------------> Echo command mode (E1)
 |       |  |  |-----------------------> Result codes displayed (Q0)
 |       |  |--------------------------> Verbal messages (V1)
 |       |-----------------------------> Normal DTR
 |-------------------------------------> Modem is DCE
```

This modem is dialed using the **DTR*** option.

This modem may be auto-dialed only if **S13** has been set to **8** and a phone number loaded into the **&Z0** register. Use the "**DTR***" option with **CONNECT=DIAL** or the **MODEM** keyword in the **RJECLINE** file. Note that the "*" appended to "DTR" will suppress the console request normally associated with DTR dialing.

USRobotics Courier V.34

If you've attached a US Robotics Courier Dual Standard V.34 or V.34 modem to the Network Engine, first make sure that the modem is correctly configured before starting. Set up the modem as follows:

Before configuration, the modem's DIP switches should be set as follows:

```
QUAD     1  2  3  4  5  6  7  8  9  10
 U       U  U  D  U  ?  D  D  D  D  D    (U=Off or Open; D=On or Closed)
 |       |  |  |  |  |  |  |  |  |  |--> Load Factory defaults on power-up
 |       |  |  |  |  |  |  |  |  |-----> Don't hangup on +++
 |       |  |  |  |  |  |  |  |--------> Smart mode
 |       |  |  |  |  |  |  |-----------> No result codes in Answer mode
 |       |  |  |  |  |  |--------------> DCD forced on pin 8
 |       |  |  |  |  |-----------------> Auto answer (U=Yes, D=No)
 |       |  |  |  |--------------------> Echo command mode (E1)
 |       |  |  |-----------------------> Result codes displayed (Q0)
 |       |  |--------------------------> Verbal messages (V1)
 |       |-----------------------------> Normal DTR
 |-------------------------------------> Modem is DCE
```

Then, enter the following "AT" commands via an ASCII terminal connected to the modem's DTE port. Set the terminal for 9600 baud, 8 data bits and no parity for this operation.

```
AT&F                  Reset to factory defaults
AT&X0                 Internal Transmit Clock
AT&M6                 Synchronous V.25 dialer
AT&N6                 9600 link speed
AT&R0                 CTS follows RTS
ATS0=0|1|...|n        0 disables auto-answer, ring count otherwise
ATS26=0               RTS-CTS delay = 0ms
AT&S1                 Normal DSR
AT&C0                 DCD forced high
AT&D2                 DTE controls DTR
ATX7                  Full detail on messages
AT&W                  Save configuration in NVRAM
```

After configuration, change DIP switch number ten (10) back to the Up (Closed) position. This will cause the modem to load its configuration from the NVRAM settings upon reset. Then, reset the modem by turning it off, then on again.

Note that if you ever need to reconfigure the modem, you will need to change DIP switch 10 back to the Down (Open) position and again reset the modem. Unfortunately, you will then have to completely re-enter the above configuration into the modem.

This modem is dialed using the **V.25,ASCII** option. This option may be specified with the **CONNECT=DIAL** sequence or the **MODEM** keyword in the **RJECLINE** file.

Racal/Vadic 9642 PA

If you've attached a Racal/Vadic 9642 PA modem to the Network Engine, first make sure that the modem is correctly configured before starting.

The key-stroke sequence to configure this modem follows. The keys on the front of the modem are labeled:

```
Key          Meaning
<-           Left Arrow
1            Option #1
2            Option #2
3            Option #3
->           Right Arrow
Ent          Enter
```

First, ensure that the modem has been reset to its factory defaults. Do this by holding the Right Arrow down while powering up the modem. Hold the Right Arrow button down until the words "FACTORY DEFAULT" appear on the display.

The key-stroke sequence is:

Key(s)	Menu (as it appears on the LCD display)
ENT	MAIN MENU <1>
2	QUICK SETUP <1>
-> ->	QUICK SETUP "V25 DIAL (Syn_c)"
2	"V25 DIAL (Syn_c)" Blinks
ENT	SETUP <1>
-> -> ->	SETUP <4>
2	EIA <1>
1	CTS to EIA
2	RTS Blinks
ENT	EIA <1>
2	CD to EIA
2	TRUE Blinks
ENT	EIA <1>
3	DSR to EIA
1	NORM Blinks
ENT	EIA <1>
ENT	SETUP <4>
3	MODEM
<-	MODEM SETUPS <8>
3	RTS-CTS - should display "0ms"
3	INC -> 15ms
3	INC -> 50ms
ENT	MODEM SETUPS <8>
ENT	SETUP <4>
<- <- <-	SETUP <1>
3	SPEED LIMIT <1>
->	SPEED LIMIT <2>
1	SPEED LIMIT TYPE
1	V.32 Blinks
	...OR...

```
3                                       208 Blinks
ENT                                     SPEED LIMIT
ENT                                     SETUP
ENT                                     Modem resets and goes back to "idle"
```

This modem is dialed using the **V.25,ASCII** option. This option may be specified with the **CONNECT=DIAL** sequence or the **MODEM** keyword in the **RJECLINE** file.

Racal/Milgo RMD 3222

If you've attached a Racal/Milgo RMD 3222 modem to the Network Engine, make sure that the modem is correctly configured before starting.

The key-stroke sequence to configure this modem follows. The keys on the front of the modem are labeled:

```
Key          Meaning
<-           Left Arrow
1            Option #1
2            Option #2
3            Option #3
->           Right Arrow
Ent          Enter
```

First, ensure that the modem has been reset to its factory defaults. Do this by holding the Right Arrow down while powering up the modem. Hold the Right Arrow button down until the words "FACTORY DEFAULT" appear on the display.

The key-stroke sequence is:

Key(s)	Menu (as it appears on the LCD display)
ENT	MAIN MENU <1>
2	QUICK SETUP <1>
-> ->	QUICK SETUP "V25 DIAL (Syn_c)"
2	"V25 DIAL (Syn_c)" Blinks
ENT	SETUP <1>
-> -> ->	SETUP <4>
2	EIA <1>
1	CTS to EIA
2	RTS Blinks
ENT	EIA <1>
2	CD to EIA
2	TRUE Blinks
ENT	EIA <1>
3	DSR to EIA
1	NORM Blinks
ENT	EIA <1>
ENT	SETUP <4>
3	MODEM
<-	MODEM SETUPS <8>
3	RTS-CTS - should display "0ms"
3	INC -> 15ms
3	INC -> 50ms
ENT	MODEM SETUPS <8>
ENT	SETUP <4>
<- <- <-	SETUP <1>
3	SPEED LIMIT <1>
->	SPEED LIMIT <2>
1	SPEED LIMIT TYPE
1	V.32 Blinks
	...OR...

```
3                                       208 Blinks
ENT                                     SPEED LIMIT
ENT                                     SETUP
ENT                                     Modem resets and goes back to "idle"
```

This modem is dialed using the **V.25,ASCII** option. This option may be specified with the **CONNECT=DIAL** sequence or the **MODEM** keyword in the **RJECLINE** file.

Racal/Milgo RMD 3296

If you've attached a Racal/Milgo RMD 3296 modem to the Network Engine, make sure that the modem is correctly configured before starting.

The key-stroke sequence to configure this modem follows. The keys on the front of the modem are labeled:

```
Key          Meaning
<-           Left Arrow
1            Option #1
2            Option #2
3            Option #3
->           Right Arrow
Ent          Enter
```

First, ensure that the modem has been reset to its factory defaults. Do this by holding the Right Arrow down while powering up the modem. Hold the Right Arrow button down until the words "FACTORY DEFAULT" appear on the display.

The key-stroke sequence is:

```
Key(s)                          Menu (as it appears on the LCD display)
ENT                             MAIN MENU <1>
2                               QUICK SETUP <1>
-> ->                           QUICK SETUP  "V25 DIAL (Syn_c)"
2                               "V25 DIAL (Syn_c)" Blinks
ENT                             SETUP <1>
-> -> ->                        SETUP <4>
2                               EIA <1>
1                               CTS to EIA
2                               RTS Blinks
ENT                             EIA <1>
2                               CD to EIA
2                               TRUE Blinks
ENT                             EIA <1>
3                               DSR to EIA
1                               NORM Blinks
ENT                             EIA <1>
ENT                             SETUP <4>
3                               MODEM
<-                              MODEM SETUPS <8>
3                               RTS-CTS - should display "0ms"
3                               INC -> 15ms
3                               INC -> 50ms
ENT                             MODEM SETUPS <8>
ENT                             SETUP <4>
<- <- <-                        SETUP <1>
ENT                             Modem resets and goes back to "idle"
```

This modem is dialed using the **V.25,ASCII** option. This option may be specified with the **CONNECT=DIAL** sequence or the **MODEM** keyword in the **RJECLINE** file.

Penril Alliance V.32/14.4M

If you've attached a Penril Alliance modem to the Network Engine, make sure that the modem is correctly configured before starting.

The key-stroke sequence to configure this modem follows. The keys on the front of the modem are labeled:

```
Key          Meaning
<-           Left Arrow
1            Option #1
2            Option #2
3            Option #3
->           Right Arrow
Ent          Enter
```

First, ensure that the modem has been reset to its factory defaults. Do this by holding the Right Arrow down while powering up the modem. Hold the Right Arrow button down until the words "FACTORY DEFAULT" appear on the display.

The key-stroke sequence is:

Key(s)	Menu (as it appears on the LCD display)
ENT	MAIN MENU <1>
2	QUICK SETUP <1>
-> ->	QUICK SETUP "V25 DIAL (Syn_c)"
2	"V25 DIAL (Syn_c)" Blinks
ENT	SETUP <1>
-> -> ->	SETUP <4>
2	EIA <1>
1	CTS to EIA
2	RTS Blinks
ENT	EIA <1>
2	CD to EIA
2	TRUE Blinks
ENT	EIA <1>
3	DSR to EIA
1	NORM Blinks
ENT	EIA <1>
ENT	SETUP <4>
3	MODEM
<-	MODEM SETUPS <8>
3	RTS-CTS - should display "0ms"
3	INC -> 15ms
3	INC -> 50ms
ENT	MODEM SETUPS <8>
ENT	SETUP <4>
<- <- <-	SETUP <1>
3	SPEED LIMIT <1>
->	SPEED LIMIT <2>
1	SPEED LIMIT TYPE
1	V.32 Blinks
	...OR...

```
3                                       208 Blinks
ENT                                     SPEED LIMIT
ENT                                     SETUP
ENT                                     Modem resets and goes back to "idle"
```

This modem is dialed using the **V.25,ASCII** option. This option may be specified with the **CONNECT=DIAL** sequence or the **MODEM** keyword in the **RJECLINE** file.

Octocom OSI 8396

The Octocom OSI 8396 is configured using the buttons on the front panel of the modem. A terminal must also be connected to the modem to enter configuration information.

Reset the modem to factory defaults by:

- Making sure the buttons labeled ANS/4, TP/2, and LLB/1 are all in the out position.
- Press the DATA button until the MR light flashes twice.

Connect a terminal to the DTE Interface on the back of the modem using a 25-pin straight-through cable. Type AT and <cr>. You should receive an OK back from the modem. If you do NOT receive the OK message back from the modem, reset the modem to factory defaults again and check the terminal's data communications configuration.

Once you receive the OK back from the modem, enter one of the following command sequences:

- If you want the modem to auto-detect V.32 (9600bps) and Bell 208 (4800bps), enter the following commands:

```
AT&F&M2&C3&D2S57=255%H2*V2%S19&W
```

- If you want the modem to auto-detect V.32 (9600bps) and Bell 201 (2400bps), enter the following commands:

```
AT&F&M2&C3&D2S57=255%H2*V2%S17&W
```

Leaving the 25-pin straight-through cable plugged into the modem, disconnect the cable from the terminal and connect the cable to the modem port of the Network Engine.

This modem is dialed using the **V.25,ASCII** option. This option may be specified with the **CONNECT=DIAL** sequence or the **MODEM** keyword in the **RJECLINE** file. Place a "T" before the phone number for tone (instead of pulse or rotary) dialing. e.g.

```
#RJLINE;CONNECT=DIAL,"T95551212",V.25,ASCII;...
```

Other Modems

If you are attempting to use a modem not mentioned above, you will need to establish that the modem is minimally configured for use with the Network Engine.

For most modems, the following capabilities should be set:

- Normal DTR. The DTE device (Network Engine) controls Data-Terminal-Ready (DTR), raising it to enable the modem and lowering it to disable the modem and break an active connection.
- Normal DSR. The DCE device (modem) controls Data-Set-Ready (DSR), raising the signal when a connection has been established and lowering the signal when a connection is broken. This signal must NOT be forced on!
- DCD. The Network Engine requires that Data-Carrier-Detect (DCD) be high in order to receive inbound packets of information. In half-duplex communications, DCD is raised in response to the remote modem's having raised Request-To-Send (RTS.) In full-duplex communications, DCD is usually on at all times. If it isn't clear how your modem treats DCD, set it On, or True.
- CTS. The Network Engine requires that Clear-To-Send (CTS) be high in order to transmit outbound packets of information. In half-duplex communications, CTS is raised by your modem in response to the Engine's having raised RTS locally. In full-duplex communications, CTS is usually on at all times. If it isn't clear how your modem treats CTS, set it On, or True; otherwise set CTS to follow RTS.
- RTS-CTS delay. For use with half-duplex modems only. If CTS is set to follow RTS, you may need to set the RTS-CTS delay parameter. This option, usually specified in milli-seconds (ms), indicates the delay between the Engine's raising RTS to the modem and the modem's raising CTS back to the Engine. The delay value should be greater than zero, 50ms at least. If this value is too short, the remote system might not receive any data from the Engine. The only disadvantage to setting the delay to too large a value will be to unnecessarily increase the connect time.
- Transmitter clock - Internal. The modem MUST provide the synchronous clock signals.

For modems with V.25, and/or V.25bis, capability, the following options should also be set:

- Enable V.25 dialer. Most V.25 capable modems are not factory configured for V.25 dialing - they typically start out in asynchronous, Hayes-compatible mode.
- Enable V.25 and/or V.25bis result codes. Some modems, by default, don't send result codes to the DTE. The Engine is programmed to recognize the V.25 responses and to pass them to the Network program.
- Enable Character Synchronous mode. Bit Synchronous, used in SDLC mode, will not work with the Engine. This sometimes appears as a choice between bisync mode (BSC) and Synchronous Data Link Control (SDLC.)

If it is not specified explicitly, the V.25 character set will have to be determined. Some modems use EBCDIC codes (Network/S's default) and others use ASCII. Note that the dialing character set has no bearing on the **LINECODE** parameter used for communications with the remote system.

APPENDIX D Procedure Examples

#RJIN Procedure

This procedure is called in place of the **FREAD** intrinsic, whenever the **FREAD** intrinsic would be called, once for each logical record in the input data file. The **#RJIN** procedure behaves as a function, returning the length of the **BUFFER** parameter as its functional return. This value is specified as a positive number of bytes.

This procedure *must* set the condition code to **CCG** in order to signify the end of the data set. This is the same condition code set by the **FREAD** intrinsic when end-of-file has been encountered.

Normally, the **CCL** condition code is ignored (to maintain compatibility with HP's RJE package.) If the **CCODE** option is set on the **#RJLINE** command line, Network/S will interpret a condition code of **CCL** as an abnormal return from the **#RJIN** procedure and will generate an appropriate error condition.

Compatibility Mode:

In SPL/V notation, the **#RJIN** procedure header would appear as follows:

```
integer procedure RJ'IN(L'BUFFER);
   logical array
      L'BUFFER;
```

In C notation, the **#RJIN** procedure header would appear as follows:

```
short int RJ_IN(BUFFER)
   char
      *BUFFER;
```

In PASCAL notation, the **#RJIN** procedure header would appear as follows:

```
procedure RJ_IN(
      var BUFFER : packed array [0..511] of char;
               ) : shortint;
```

Network/S allocates ten words of global storage for use by the **#RJIN** procedure (as well as the **#RJSTAT**, **#RJOUT** and **#RJCONTINUE** procedures.)

The words are allocated at memory locations DB+0 through DB+9 and are initialized to zeroes when Network/S is first run. Network/S will not modify these locations after initialization, so their values can be used between calls to the user procedures.

Native Mode:

In SPLash! notation, the **#RJIN** procedure header would appear as follows:

```
integer procedure RJ'IN(L'BUFFER);
   logical array
      L'BUFFER;
   option
      native;
```

The C/iX specification matches the C specification for compatibility mode.

The PASCAL/XL specification matches the PASCAL specification for compatibility mode.

Note that the Native Mode version of Network/S does *not* allocate any global memory for use by the XL procedures. Native Mode library procedures may declare permanent storage on their own.

#RJIN Procedure Example

The following Example section contains the SPL/V source for a sample compatibility mode **#RJIN** procedure. It attempts to open a file back-referenced by **:FILE RJIN=...** and then returns one logical record per call. The procedure will also return the condition code returned from the **FREAD** intrinsic back to Network/S.

```
$control uslinit,subprogram,segment=RJE'PROCEDURES
begin

   integer procedure RJIN(L'BUF);

      logical array
         L'BUF;

      begin

         equate
            N'PARMS         = 1,
            CCG             = 0,
            CCL             = 1,
            CCE             = 2;
```

```
        define
           CCODE           = 6:2#;
        byte array
           FILE'NAME(0:29);
        integer
           FILE'NUM        = DB+0,
           REC'SIZE        = DB+1;
        logical
           RETURN'STATUS   = Q-1;
        intrinsic
           FOPEN,
           FFILEINFO,
           FREAD,
           FCLOSE,
           PRINTFILEINFO,
           QUIT;

        subroutine ABEND(CC,LOC);
           value         CC,LOC;
           integer       CC,LOC;
           begin
              PRINTFILEINFO(FILE'NUM);
              RETURN'STATUS.(CCODE):=CC;
              assemble(EXIT N'PARMS);  ! If CCODE=YES on #RJLINE
              QUIT(LOC)                ! If CCODE=NO on #RJLINE
           end;  <<ABEND>>

        if (FILE'NUM = 0) then
        begin  ! First time - perform initialization
           move FILE'NAME:="*RJIN ";
           FILE'NUM:=FOPEN(FILE'NAME,%002005,%000000);
           if < then
              ABEND(CCL,1);
           FFILEINFO(FILE'NUM,4,REC'SIZE);
           if < then
              ABEND(CCL,2)  ! Fatal FFILEINFO error
        end;
        RJIN:=FREAD(FILE'NUM,L'BUF,REC'SIZE);
        if < then
           ABEND(CCL,3);  ! Fatal FREAD error
        if > then
        begin  ! End of file - close file and return CCG
           RETURN'STATUS.(CCODE):=CCG;
           FCLOSE(FILE'NUM,0,0);
           FILE'NUM:=0
        end
        else   ! Successful read - return CCE
           RETURN'STATUS.(CCODE):=CCE

     end;  <<RJIN>>
end.
```

#RJOUT Procedure

This procedure is called in place of the **FWRITE** intrinsic, whenever the **FWRITE** intrinsic would be called, once for each logical record in the output data set.

Normally, no condition code checking is performed after the call to the **#RJOUT** procedure, (to maintain compatibility with HP's RJE package.) If the **CCODE** option is set on the **#RJLINE** command line, Network/S will interpret a condition code of **CCL** or **CCG** as an abnormal return from the **#RJOUT** procedure and will generate an appropriate error condition.

Compatibility Mode:

In SPL/V notation, the **#RJOUT** procedure header would appear as follows:

```
procedure RJ'OUT(L'BUFFER,BUF'LEN);
   value
      BUF'LEN;
   logical array
      L'BUFFER;
   integer
      BUF'LEN;
```

In C notation, the **#RJOUT** procedure header would appear as follows:

```
void RJ_OUT(BUFFER,BUF_LEN)
   char
      *BUFFER;
   short int
      BUF_LEN;
```

In PASCAL notation, the **#RJOUT** procedure header would appear as follows:

```
procedure RJ_OUT(
      var BUFFER  : packed array [0..511] of char;
          BUF_LEN : shortint
                );
```

Two parameters are passed to the procedure using the normal techniques, the address of the buffer containing the data to be written and a value indicating the length of the buffer (a positive value indicates a word count and a negative value indicates a byte count.)

Two additional items are made available to the procedure, a carriage control parameter and an end-of-file flag. These values are “pushed” onto the stack prior to the procedure call and are available in the memory locations identified by:

```
CCTL     -  Q-6
EOF      -  Q-7
```

The **CCTL** parameter contains the carriage control directive that would have been used with the **FWRITE** intrinsic. The **EOF** flag is set in the final call to the procedure, which is made after Network/S has received an End-of-Transmission (**EOT**) code from the remote system.

These parameters are accessible only from an SPL/V **#RJOUT** procedure. They *must not* be specified in the procedure header. It is the procedure's responsibility to “pop” formal parameters from the stack. These two parameters are explicitly “pushed” and “popped” by Network/S - if your procedure treats them as formal parameters, popping them before returning to Network/S, a stack underflow will result!

Network/S allocates ten words of global storage for use by the **#RJOUT** procedure (as well as the **#RJIN, #RJSTAT** and **#RJCONTINUE** procedures.)

The words are allocated at memory locations DB+0 through DB+9 and are initialized to zeroes when Network/S is first run. Network/S will not modify these locations after initialization, so their values can be used between calls to the user procedures.

Native Mode:

In SPLash! notation, the **#RJOUT** procedure header would appear as follows:

```
procedure RJ'OUT(L'BUFFER,BUF'LEN,CCTL,EOF);
   value
      BUF'LEN,
      CCTL,
      EOF;
   logical array
      L'BUFFER;
   integer
      BUF'LEN;
   logical
      CCTL,
      EOF;
   option
      native;
```

In C notation, the **#RJOUT** procedure header would appear as follows:

```
void RJ_OUT(BUFFER,BUF_LEN,CCTL,EOF)
   char
      *BUFFER;
   short int
      BUF_LEN,
      CCTL,
      EOF;
```

In PASCAL/XL notation, the **#RJOUT** procedure header would appear as follows:

```
procedure RJ_OUT(
      var BUFFER  : packed array [0..511] of char;
          BUF_LEN : shortint;
          CCTL    : shortint;
```

```
                EOF     : shortint
              );
```

Note that with the Native Mode interface, the **CCTL** and **EOF** parameters are passed in the argument list of the procedure.

Note also that the Native Mode version of Network/S does *not* allocate any global memory for use by the XL procedures. Native Mode library procedures may declare permanent storage on their own.

#RJOUT Procedure Example

The following Example section contains the SPL/V source for a sample compatibility mode **#RJOUT** procedure. It attempts to open a file back-referenced by **:FILE RJOUT=...** and then writes each received record to that file. The procedure assumes that Network/S has stacked two additional parameters (at Q-6 and Q-7 as with HP's RJE) which contain the carriage control code to be used and a Boolean end-of-file indicator.

```
$control uslinit,subprogram,segment=RJE'PROCEDURES
begin

   procedure RJOUT(L'BUFFER,BUF'LEN);

      value
         BUF'LEN;
      logical array
         L'BUFFER;
      integer
         BUF'LEN;

      begin

         equate
            N'PARMS         = 2,
            CCG             = 0,
            CCL             = 1,
            CCE             = 2;
         define
            CCODE           = 6:2#;
         integer
            FILE'NUM        = DB+0;
         logical
            EOF             = Q-7,
            CCTL            = Q-6,
            RETURN'STATUS   = Q-1;
         byte array
            FILE'NAME(0:29);
         intrinsic
            FOPEN,
            FWRITE,
            FCLOSE,
            PRINTFILEINFO,
            QUIT;

         subroutine ABEND(CC,LOC);
            value         CC,LOC;
            integer       CC,LOC;
            begin
               PRINTFILEINFO(FILE'NUM);
               RETURN'STATUS.(CCODE):=CC;
               assemble(EXIT N'PARMS);  ! if CCODE=YES on #RJLINE
               QUIT(LOC)                ! if CCODE=NO on #RJLINE
            end;  <<ABEND>>
```

```
         if (FILE'NUM = 0) then
         begin
            move FILE'NAME:="*RJOUT ";
            FILE'NUM:=FOPEN(FILE'NAME,%002005,%000001);
            if < then
               ABEND(CCL,1)  ! Fatal FOPEN error
         end
         else if EOF then
         begin  ! End of file - close file
            FCLOSE(FILE'NUM,0,0);
            FILE'NUM:=0
         end;
         if (FILE'NUM <> 0) then
         begin
            FWRITE(FILE'NUM,L'BUFFER,BUF'LEN,CCTL);
            if <> then
               ABEND(CCL,2)  ! Fatal FWRITE error
         end

      end;  <<RJOUT>>

end.
```

#RJSTAT Procedure

This procedure is called in place of the **FWRITE** intrinsic, whenever the **FWRITE** intrinsic would be called, once for each line in the status display.

Normally, no condition code checking is performed after the call to the **#RJSTAT** procedure, (to maintain compatibility with HP's RJE package.) If the **CCODE** option is set on the **#RJLINE** command line, Network/S will interpret a condition code of **CCL** or **CCG** as an abnormal return from the **#RJSTAT** procedure and will generate an appropriate error condition.

Compatibility Mode:

In SPL/V notation, the **#RJSTAT** procedure header would appear as follows:

```
procedure RJ'STAT(L'BUFFER,BUF'LEN);
   value
      BUF'LEN;
   logical array
      L'BUFFER;
   integer
      BUF'LEN;
```

In C notation, the **#RJSTAT** procedure header would appear as follows:

```
void RJ_STAT(BUFFER,BUF_LEN)
   char
      *BUFFER;
   short int
      BUF_LEN;
```

In PASCAL notation, the **#RJSTAT** procedure header would appear as follows:

```
procedure RJ_STAT(
      var BUFFER  : packed array [0..511] of char;
          BUF_LEN : shortint
                 );
```

Two parameters are passed to the procedure using the normal techniques, the address of the buffer containing the data to be written and a value indicating the length of the buffer (a positive value indicates a word count and a negative value indicates a byte count.)

One additional item is made available to the procedure, a carriage control parameter. This value is “pushed” onto the stack prior to the procedure call and is available in the memory location identified by:

```
CCTL     -  Q-6
```

The **CCTL** parameter contains the carriage control directive that would have been used with the **FWRITE** intrinsic.

This parameter is accessible only from an SPL/V **#RJSTAT** procedure.

Network/S allocates ten words of global storage for use by the **#RJSTAT** procedure (as well as the **#RJIN, #RJOUT** and **#RJCONTINUE** procedures.)

The words are allocated at memory locations DB+0 through DB+9 and are initialized to zeroes when Network/S is first run. Network/S will not modify these locations after initialization, so their values can be used between calls to the user procedures.

Native Mode:

In SPLash! notation, the **#RJSTAT** procedure header would appear as follows:

```
procedure RJ'STAT(L'BUFFER,BUF'LEN,CCTL);
   value
      BUF'LEN,
      CCTL;
   logical array
      L'BUFFER;
   integer
      BUF'LEN;
   logical
      CCTL;
   option
      native;
```

In C/iX notation, the **#RJSTAT** procedure header would appear as follows:

```
void RJ_STAT(BUFFER,BUF_LEN,CCTL)
   char
      *BUFFER;
   short int
      BUF_LEN,
      CCTL;
```

In PASCAL/iX notation, the **#RJSTAT** procedure header would appear as follows:

```
procedure RJ_STAT(
      var BUFFER  : packed array [0..511] of char;
          BUF_LEN : shortint;
          CCTL    : shortint
                 );
```

Note that with the Native Mode interface, the **CCTL** parameter is passed in the argument list of the procedure.

Note also that the Native Mode version of Network/S does *not* allocate any global memory for use by the XL procedures. Native Mode library procedures may declare permanent storage on their own.

#RJSTAT Procedure Example

The following Example section contains the SPL/V source for a sample compatibility mode **#RJSTAT** procedure. It simply displays each statistic line to **$STDLIST** as it is received (exactly as the **#RJSTAT** command behaves by default.)

```
$control uslinit,subprogram,segment=RJE'PROCEDURES
begin

   procedure RJSTAT(TARGET,COUNT);

      value
         COUNT;
      logical array
         TARGET;
      integer
         COUNT;

      begin

         logical
            CCTL            = Q-6;
         intrinsic
            PRINT;

         PRINT(TARGET,COUNT,CCTL)

      end;  <<RJSTAT>>

end.
```

#RJCONTINUE Procedure

Compatibility Mode:

In SPL/V notation, the **#RJCONTINUE** procedure header would appear as follows:

```
procedure RJ'CONTINUE(FILE'NUM,ERRORS,CMD,NEW'FILE,ACTION);
   integer
      FILE'NUM;
   integer array
      ERRORS;
   byte array
      CMD,
      NEW'FILE;
   integer
      ACTION;
```

In C notation, the **#RJCONTINUE** procedure header would appear as follows:

```
void RJ_CONTINUE(FILE_NUM,ERRORS,CMD,NEW_FILE,ACTION)
   short int
      *FILE_NUM,
      ERRORS[];
   char
      *CMD,
      *NEW_FILE;
   short int
      *ACTION;
```

In PASCAL notation, the **#RJCONTINUE** procedure header would appear as follows:

```
procedure RJ_CONTINUE(
      var FILE_NUM : shortint;
      var ERRORS   : array [0..2] of shortint;
      var CMD      : packed array [0..255] of char;
      var NEW_FILE : packed array [0..39] of char;
      var ACTION   : shortint
                     );
```

Network/S allocates ten words of global storage for use by the **#RJCONTINUE** procedure (as well as the **#RJSTAT, #RJOUT** and **#RJIN** procedures.)

The words are allocated at memory locations DB+0 through DB+9 and are initialized to zeroes when Network/S is first run. Network/S will not modify these locations after initialization, so their values can be used between calls to the user procedures.

Native Mode:

In SPLash! notation, the **#RJCONTINUE** procedure header would appear as follows:

```
procedure RJ'CONTINUE(FILE'NUM,ERRORS,CMD,NEW'FILE,ACTION);
   integer
      FILE'NUM;
   integer array
      ERRORS;
   byte array
      CMD,
      NEW'FILE;
   integer
      ACTION;
   option
      native;
```

The C/iX notation matches the C notation for compatibility mode.

The PASCAL/XL notation matches the PASCAL notation for compatibility mode.

Note that the Native Mode version of Network/S does *not* allocate any global memory for use by the XL procedures. Native Mode library procedures may declare permanent storage on their own.

#RJCONTINUE Procedure Example

The following Example section contains the SPL/V source for a sample compatibility mode **#RJCONTINUE** procedure. It displays all pertinent information passed to it by Network/S and then prompts the user for the Action code (and any related information) before returning to Network/S.

```
$control uslinit,subprogram,segment=RJE'PROCEDURES
begin

   procedure RJ'CONTINUE(FILE'NUM,ERRORS,COM'IMAGE,NEW'CMDFILE,ACTION);

      integer
         FILE'NUM,             ! MPE file number of command file
         ACTION;               ! Return 1-5
      integer array
         ERRORS;               ! Input error information
      byte array
         COM'IMAGE,            ! Current command
         NEW'CMDFILE;          ! New command file (ACTION = 2)

      begin

         equate
            CR              = %015,
            CR'LF           = %040,
            NO'CCTL         = %320;
         logical array
            L'MSG(0:128);
         byte array
            MSG(*)          = L'MSG;
```

```
         byte pointer
            PTR:=-1;
         integer
            LEN:=0;
         intrinsic
            ASCII,
            BINARY,
            FFILEINFO,
            PRINT,
            READX;

<<** Announce entry to RJCONTINUE procedure **>>
         MSG:="-"; move MSG(1):=MSG,(78);
         move MSG(10):="#RJCONTINUE error handler";
         PRINT(L'MSG,-79,CR'LF);

<<** Print the name of the current command file **>>
         move MSG:="Command file currently being read: ";
         PRINT(L'MSG,-35,NO'CCTL);
         FFILEINFO(FILE'NUM,1,MSG);
         LEN:=scan MSG until " ";
         PRINT(L'MSG,-LEN,CR'LF);

<<** Display the current command (the one that just failed) **>>
         move MSG:="Command which caused the error:";
         PRINT(L'MSG,-31,CR'LF);
         LEN:=scan COM'IMAGE until CR;
         move MSG:=COM'IMAGE,(LEN),2;
         PRINT(L'MSG,-LEN,CR'LF);

<<** Display the three-word error information array **>>
         move MSG:="***ERRORS(",2; @PTR:=tos;
         move PTR(ASCII(ERRORS,10,PTR)):="[",2;
         case ERRORS of
         begin
       <<0>> move *:="FS",2;
       <<1>> move *:="CS",2;
       <<2>> move *:="LINE",2;
       <<3>> move *:="RIN",2;
       <<4>> move *:="PROC",2;
       <<5>> move *:="CMD",2;
       <<6>> move *:="ROUTE",2;
       <<7>> move *:="SYNTAX",2;
       <<8>> move *:="Telamon",2;
            end;
            move *:="],",2; @PTR:=tos;
            tos:=@PTR(ASCII(ERRORS(1),10,PTR));
            case ERRORS of
            begin
               case ERRORS(1) of                ! 0
               begin
                  move *:="[List]",2;
                  move *:="[Punch]",2;
                  move *:="[Output]",2;
```

```
      move *:="[Command]",2;
      move *:="[Input]",2;
      move *:="[Source]",2;
      move *:="[Message]",2;
      move *:="[Statistic]",2;
      ;
      ;
      move *:="[Trace]",2;
      move *:="[Network Engine]",2;
      move *:="[RJECOM]",2;
   end;
   case ERRORS(1) of                ! 1
   begin
      move *:="[COPEN]",2;
      move *:="[CREAD]",2;
      move *:="[CWRITE]",2;
      move *:="[CCONTROL]",2;
   end;
   ;                                ! 2
   ;                                ! 3
   begin                            ! 4
      move PTR:="[LOAD ERR#",2; @PTR:=tos;
      move PTR(ASCII(ERRORS(1),10,PTR)):="]",2
   end;
   ;                                ! 5
   ;                                ! 6
   ;                                ! 7
   ;                                ! 8
end;
move *:=",",2; @PTR:=tos;
@PTR:=@PTR(ASCII(ERRORS(2),10,PTR));
if (ERRORS = 0) then
begin
   move PTR:="[FSERR ",2; @PTR:=tos;
   move PTR(ASCII(ERRORS(2),10,PTR)):="]",2;
   @PTR:=tos
end
else if (ERRORS = 1) then
begin
   if (ERRORS(2).(0:8) <> 0) then
   begin
      move PTR:="[CSWARN ",2; @PTR:=tos;
      move PTR(ASCII(ERRORS(2).(0:8),10,PTR)):="]",2;
      @PTR:=tos
   end;
   if (ERRORS(2).(8:8) <> 0) then
   begin
      move PTR:="[CSERROR ",2; @PTR:=tos;
      move PTR(ASCII(ERRORS(2).(8:8),10,PTR)):="]",2;
      @PTR:=tos
   end
end;
move PTR:=")***",2;
LEN:=tos-logical(@MSG);
```

```
         PRINT(L'MSG,-LEN,CR'LF);

<<** Get the action code to be returned to Network/S **>>
         ACTION:=0;
         do begin
            move MSG:="Action?";
            PRINT(L'MSG,-7,NO'CCTL);
            if ((ACTION:=BINARY(MSG,READX(L'MSG,-8))) = 1) then
            begin
               move MSG:="---Retry failed operation";
               PRINT(L'MSG,-25,CR'LF)
            end
            else if (ACTION = 2) then
            begin
               move MSG:="---Branch to a new command file";
               PRINT(L'MSG,-31,CR'LF);
               move MSG:="What is the new command file?";
               PRINT(L'MSG,-29,NO'CCTL);
               MSG(LEN:=READX(L'MSG,-40)):=" ";
               move NEW'CMDFILE:=MSG,(LEN+1)
            end
            else if (ACTION = 3) then
            begin
               move MSG:="---Do the next command in this command file";
               PRINT(L'MSG,-43,CR'LF)
            end
            else if (ACTION = 4) then
            begin
               move MSG:="---Exit this command file";
               PRINT(L'MSG,-25,CR'LF)
            end
            else if (ACTION = 5) then
            begin
               move MSG:="---Execute a new command";
               PRINT(L'MSG,-24,CR'LF);
               move MSG:="What is the new command?";
               PRINT(L'MSG,-24,NO'CCTL);
               MSG(LEN:=READX(L'MSG,-256)):=CR;
               move COM'IMAGE:=MSG,(LEN+1)
            end
            else
            begin
               ACTION:=-1;
               move MSG:="***Invalid ACTION - must be from 1 to 5***";
               PRINT(L'MSG,-42,CR'LF)
            end
         end until (ACTION <> 0);
         MSG:="-"; move MSG(1):=MSG,(78);
         move MSG(10):="Returning to program";
         PRINT(L'MSG,-79,CR'LF)

      end;  <<RJ'CONTINUE>>

end.
```

#RJCONTINUE Parameters

Command File Number

The first parameter contains the file number of the currently opened command file. The file may be accessed using normal file system intrinsics, and since it is passed by reference, the file number may be changed.

Error Code Array

The error code array identifies the specific error that has occurred. The first element in the array identifies the type of error. Its values can be:

```
0          MPE File System error
1          Communications (CS) error
2          Line Error (not used by Network/S)
3          RIN Error
4          Procedure Error
5          Command Error
6          Routing Error
7          Syntax Error
```

Telamon's extensions to this list include:

```
8          Network/S error
9          MPE Command error
```

The second and third elements' meanings vary according to the type of error indicated in the first element. The following table summarizes the sets of values. Most of the values shown are identical to those found in HP's RJE/3000 Reference Manual.

TABLE D-1.

Error(0)	Error(1)	Error(2)
0 - MPE File System	0 - List file	Actual MPE error number
	1 - Punch file	
	2.- Output file	
	3 - Command file	
	4 - Input file	
	6 - Message file	
	7 - Statistics file	
	8 - Temporary STDIN file	
	9 - Translate TABLE file	
	10 - TRACE file	
	11 - DEV file	
	12 - Default command file	
1 - Communications	0 - Open (RJLINE)	CS Error number
	1 - Read (RJOUT)	
	2 - Write (RJIN)	
2 - Line Error	0 - No line specified	{not used}
3 - RIN Error	0 - Invalid RIN value	{not used}
4 - Procedure Error	# - LOADPROC error	{not used}
5 - Command Error	0 - Invalid command	{not used}
6 - Routing Error	0 - List during #RJPUNCH	{not used}
	1 - Punch during #RJLIST	{not used}
7 - Syntax Error	1 - Duplicate Keyword	Byte offset
	3 - Invalid Parameter	Byte offset
	4 - Invalid numeric parm	Byte offset
8 - Network/S Error	Error Number	Other info
9 - MPE Command Error	CIERROR value	Other info

Errors of type #8 are generally internal errors that should be referred to Telamon, when and if they occur.

For testing with **#RJIF**, the three elements in the array are also stored in Job Control Words. They are:

```
ERRORCLASS          Error(0)
ERRORTYPE           Error(1)
ERRORNUM            Error(2)
```

Command Image

The third parameter is a character array, terminated with a carriage-return character (Decimal 13, Octal %15, Hex $0D.) The array contains the original text of the command being executed at the time of the error. When the Action Code, below, is set to the value **5**, this parameter is used to specify a new command to be executed by Network/S. If specified, the value must be left-justified, with a carriage-return terminator.

Command File Name

When the Action Code, below, is set to the value **2**, this parameter is used to specify the new command file to be opened and executed. The value must be left-justified, with a blank (space) terminator.

Action Code

The Action Code parameter is used by the **#RJCONTINUE** procedure to tell Network/S what to do next. Valid values are:

1 Retry the current command
2 Switch to a new command file
3 Continue with the next command in the current command file
4 Exit the current command file
5 Retry the current command, using the image passed *back* in the Command Image variable

APPENDIX E Program Messages

CS Error Messages

The first number following the CSERR keyword identifies the operation being performed when the error occurred. The possible values are:

0	line open (**#RJLINE**)
1	line read (**#RJOUT**/**#RJLIST**/**#RJPUNCH**)
2	line write (**#RJIN**)

The second number is the actual error code.

A third, optional, number may be present on **CSERR 207** errors to provide more detail on the type of transmission error.

For those similar errors that may occur at different locations within Network/S, a fourth value may be displayed in brackets ([]) to provide further information on the context of the error.

For example, **CSERR 2,103** indicates that the line was dropped during the execution of an **#RJIN** command.

CSERR 0,8

Driver not compatible with the attributes of the communications line.

Cause:

The port (LDEV) specified in the **#RJLINE** command is not attached to the Network Engine; it may be the port you are logged on to or it may not be a direct connect terminal port.

Action:

Correctly specify the port that the Engine is attached to. Make sure the port is a direct connect terminal port.

CSERR 0,11

Device not available.

Cause:

The requested port is unavailable, perhaps because it is being used by another job.

Action:

Check that someone else or some other process is not using the port that the Network Engine is connected to. Use the :SHOWDEV *ldev number* of the Network Engine port command to check the ownership of the port. i.e. #S260: 2 files. Use the **#RJLINE RIN** option to prevent this error.

CSERR x,41

Does not have autodial capability.

Cause:

The 801 auto-dial option was requested but is not supported by the hardware.

Action:

Only Network Engines with the 801 option support the 801 auto-dialer.

CSERR x,54

Auto-dialer detected errors.

Cause:

The autodialer reported an error, either a busy signal, no answer or no dial tone.

Action:

Look above this error message on **$STDLIST** for the specific call progress message that indicates the error. For V.25 auto-dialers, this message contains the string “CFI”. Look in your modem’s manual for an explanation of the error message.

CSERR x,57

Console operator replied NO to a dial prompting message.

Cause:

The manual dial attempt was unsuccessful, due to the console operator replying **No** to the console request generated by Network/S.

Action:

Find out why operator replied this way to dial request.

CSERR x,103 [1|3]

Data set not ready.

Cause:

Usually due to a data-format or password problem during transmission.

Action:

This error is caused by the remote system disconnecting the line. Contact technical support for the remote system to determine the cause. This error can be caused by incorrect passwords and/or JCL syntax and sometimes due to the inappropriate use of the **COMPRESS** and/or **TRUNCATE** options during an **#RJIN**.

CSERR x,103 [4|5|6|10]

Data set not ready.

Cause:

The remote system disconnected while Network/S was attempting to *start* a transmission. As no data has yet been sent, this error cannot be associated with the data that Network/S is currently attempting to transmit.

Action:

If this error occurs during the execution of the first **#RJIN** command, then modem may not be set correctly or linecode option may be incorrect. If an **#RJIN** worked earlier in the same transmission, check that the first **#RJIN** transmission sent the correct password (for EDI). Then, contact the remote site and find out why remote system hung up.

CSERR x,103 [7]

Data set not ready.

Cause:

The remote system went offline while there was untransmitted data in the Engine's buffer.

Check if remote host supports data compression (**COMPRESS**) and /or short line truncation (**TRUNCATE**.) Verify that format of file including password and JCL is correct.

CSERR x,103 [8|11]

Data set not ready.

Cause:

The remote system went offline while Network/S was waiting to receive a line bid.

Action:

If this **#RJOUT** is the first command following the **#RJLINE**, then the modem may not be set correctly or the **LINECODE** option may be incorrect. If an **#RJIN** worked earlier in the same transmission, check that the first **#RJIN** transmission sent the correct password (for EDI). Then, contact the remote site and find out why the remote system disconnected.

CSERR x,103 [9]

Data set not ready.

Cause:

The remote system went offline while receiving data. This is usually a modem or line problem.

Action:

Call back and re-attempt the transmission.

CSERR x,124

Reset while in RAM.

Cause:

The Engine has reported a software reset. This can have a number of causes. If your system is heavily loaded, the Network Engine may time-out during an asynchronous operation with the Network/S software. If this happens, the Engine assumes that the HP 3000 has “crashed” and resets itself to drop the line. Otherwise, this error may indicate an internal problem with the Network Engine.

Action:

Try increasing the process priority of Network/S using the **PRI=HIGH** option on the **#RJLINE** command. This will minimize the possibility that the Network Engine will time-out communicating with Network/S. Alternatively, you can increase the **CRASH** timer, specified in the **RJECLINE** file. The default value is 10 minutes, so unless you have specified a lower value, this option is probably not the cause. Contact Telamon.

CSERR x,151

Connect time-out.

Cause:

The **WAIT** timer on **#RJLINE** command expired. The modem may not have enough time to establish connection, or an incorrect dialer-type was specified on the **#RJLINE** command.

Action:

Verify the dialer specified on the **#RJLINE** command. Make the **WAIT** time longer and verify that phone number is correct. Find out why local system is not responding, perhaps by dialing the number from a voice phone and listening for a modem response. The modem may need to be reset. The modem may not have enough time between an earlier disconnect and this dial attempt.

CSERR x,154 [1|2]

Power failure.

Cause:

The Network Engine has reported a power failure, due either to the reset switch being pressed or there being a problem with the Engine's power supply

Action:

Check to see if there was a power failure, if the Network Engine had been unplugged, or if the reset switch was pressed.

CSERR x,155 [1|2]

Local time-out.

Cause:

The **WAIT** timer on an **#RJIN** expired. The RTS-CTS delay on the modem may be set too low or the remote system stopped responding due to some other problem.

Action:

Change the modem's RTS-CTS delay to a higher value (above 50ms) or find out why remote system stopped responding.

CSERR x,158

Remote terminal sent shutdown sequence and disconnected.

Cause:

Network/S received a DLE-EOT (end of transmission character) sequence at an unexpected point. As DLE-EOT indicates a switch-line disconnect, this is interpreted as an error. A number of systems send this sequence in lieu of an EOT sequence to mark the end of a transmission.

Action:

Find out why the remote system sent this sequence. If this error occurred at the end of a job, try adding the **XEND** parameter to the **#RJLINE** command. This will cause the DLE-EOT to be interpreted as an EOT instead.

CSERR x,201

Operation aborted.

Cause:

The current command has been terminated by user intervention (**Control-Y I**).

CSERR x,202

Invalid user request.

Cause:

The **#RJLINE SELECT** parameter was requested on a non-Dual Modem option Network Engine. Only Network Engines with the Dual Modem option support multiple modems.

Action:

Correct the **#RJLINE** command.

CSERR 2,205

Remote primary station bid for the line while local user was also bidding.

Cause:

A line bid was received while Network/S was waiting for acknowledgment for its own **#RJIN** line bid.

Action:

This is usually caused by having the **#RJIN** and **#RJOUT** commands in the incorrect order in your command file. The specific indication is that you are attempting to transmit data while the remote system is attempting to transmit data to *you.*

CSERR 2,206

Remote has requested to send. (An RVI sequence was received.)

Cause:

The remote system has an urgent message to send. This condition occurs while you are executing an **#RJIN** command, and indicates that the remote system needs to interrupt your transmission. Network/S will attempt to perform an **#RJOUT** $STDLIST to accept the possibly urgent message.

Action:

Change the command file and place an **#RJOUT** at the point following the **#RJIN** that failed with this error. If you are sending multiple files with consecutive **#RJIN** commands, try separating the files and send them with **#RJOUT** commands between the **#RJIN** commands.

CSERR x,207,1

Transmission retry count was exhausted.
Invalid ID sequence received.

Cause:

An invalid remote ID (**REMID**) sequence was received.

Action:

Verify the **#RJLINE REMID** value and try again.

CSERR x,207,3

Transmission retry count was exhausted.
Block check character of field check sequence error.

Cause:

This is an indication of a block check error, probably caused by line noise. The Network Engine will only attempt to send a data block 'n' times, 'n' controlled by the **NAK** option in the **RJECLINE** file.

Action:

If due to line noise, simply retry the job. You can also increase the **NAK** limit in **RJECLINE** file to increase the number of attempts that the Network Engine will make under these circumstances.

CSERR 2,207,4

Transmission retry count was exhausted.
Response time-out occurred.

Cause:

The remote stopped responding during an **#RJIN** operation.

Action:

Increase RTS-CTS delay and/or increase the **ENQ** limit in **RJECLINE** file. Check with the remote system's personnel for other indications of problems.

CSERR 2,207,7

Transmission retry count was exhausted.
Remote station did not respond to the local line bid.

Cause:

During an **#RJIN** operation, the remote system failed to respond to the initial line bid.

Action:

Increase RTS-CTS delay. Increase **ENQ** limit in **RJECLINE** file.

CSERR 1,209

Receive time-out.

Cause:

The **#RJOUT WAIT** time expired.

Action:

Increase the **WAIT** timer value and/or verify that the remote system has data to send. Verify that remote system is *not* expecting a file from you at the same time.

CSERR 2,210[1|2]

Remote terminal sent end-of-transmission.

Cause:

An **EOT** was received during the execution of an **#RJIN** command.

Action:

Find out why the remote system sent the **EOT** during transmission.

CSERR 1,217

No line bid was received from the remote causing a local time-out.

Cause:

The **#RJOUT WAIT** timer expired before a line bid was received.

Action:

#RJIN and **#RJOUT** commands may be in the wrong order (both systems may be expecting transmission at same time). Increase timer.

CSERR 1,220

An **EOT** was received from the remote station before the last block of a multiblock transmission was sent.

Cause:

With the **NEEDETX** option specified on the **#RJOUT** command, this error indicates that an **EOT** was received, but the last block in the transmission did *not* terminate with an End-of-Text (**ETX**.)

Action:

This error can occur if the remote system disconnects prematurely, sending an **EOT** prior to dropping the connection. If the **XEND** option was specified on the **#RJLINE** command, a **DLE-EOT** was sent.

CSERR 1,223

Too much data was transmitted by the remote station. Part of the data was lost. Buffer overflow.

Cause:

This error indicates that the Network Engine's internal receive buffer has overflowed. If using 3780 emulation, this means that the remote system has continued to send despite the Network Engine's **WAK**s, normally used as a flow control mechanism in these instances.

Action:

The HP 3000 may be running too slowly to keep the receive buffer from overflowing. This is often caused by having the asynchronous speed between the Network Engine and the HP 3000 set too low. When possible, set this speed to 19,200 baud, *regardless* of the modem's baud rate. Otherwise, find out if the remote system obeys the Wait Acknowledge (**WACK**) sequence for flow control

Other Error and Informative Messages

***intrinsic* error on *type* file (*info*)**
error message

This message indicates a local file system error. *intrinsic* identifies the specific intrinsic that was called and *type* identifies the type of file that was being accessed. *info*, if displayed, provides further information. For example, if an **#RJIN** is attempted on a non-existent file, the result would be:

```
#RJIN NOSUCHFL
RJIN Settings - File reference: "NOSUCHFL"
  INCODE=ASCII; XPARENT=NO; TRUNCATE=YES; COMPRESS=YES
  Effective MAXRPB=255
  WAIT=3,0
ddmmmyy-hh:mm:ss.t: FOPEN error on RJIN file (NOSUCHFL)
ddmmmyy-hh:mm:ss.t: NONEXISTENT PERMANENT FILE  (FSERR 52)
```

Under MPE/iX, it is a common occurrence get the following error executing the **#RJLINE** command:

```
#RJLINE ...
RJLINE Settings
  3780; ...
  DEV=...
ddmmmyy-hh:mm:ss.t: FOPEN error on DEV file (RJELINE)
ddmmmyy-hh:mm:ss.t: SOFTWARE ABORT  (FSERR 32)
```

This error is caused by a condition in the Distributed Terminal Controller (DTC) and can only be rectified by resetting the DTC. Contact your System Administrator/System Manager for information on resetting the DTC.

Issuing transmit bid
Transmit bid acknowledged: starting transfer
Waiting for line bid
Line bid received: starting transfer

These messages are displayed to mark the initiation of **#RJIN** and **#RJOUT** commands, respectively. The first message is displayed when **#RJIN** first starts bidding for the line and the second message is displayed once the remote system has acknowledged the bid. The third message is displayed when **#RJOUT** starts waiting for an incoming line bid and the last message is displayed once that line bid is received and acknowledged.

----------{Start|End} data set----------

When an **#RJOUT** is performed and the output file is **$STDLIST**, this message is displayed prior to the start and after the **EOT** has been received, in order to distinguish received data from Network/S messages.

Waiting for connection
On Line (DSR high)

The first message is displayed when Network/S starts waiting to go online and the second message is displayed once the Network Engine has reported that the modem device is online, with **DSR** high.

Warning: EOT not preceded by ETX

This message is displayed if an **#RJOUT** command terminates with the receipt of an **EOT**, but the last block received was not terminated with an **ETX** character. This might not be a problem, but very often indicates that the transmission ended prematurely. If the **NEEDETX** option is specified, this warning will be instead treated as CSERR 220.

Warning: EOT assumed after ETX

This message is displayed if the **TIMEOUTOK=ETX** option was specified on the **#RJOUT** command, and a time-out occurred after having received an **ETX**-terminated data block. In this case, Network/S assumes that the expected **EOT** was lost due to line problems and proceeds without error.

Warning: {GS|NAK} characters in non-xparent record

GS: when compression is performed during an **#RJIN** command, Network/S replaces groups of blank characters with two-byte sequences, consisting of the **GS** character followed by a count of the blanks compressed. If, during this operation, Network/S encounters **GS** characters in the data, this warning will be generated.

NAK: while supported in the bisync standard, the presence of Negative Acknowledge (**NAK**) characters in the data stream can cause problems with some host systems, sometimes with unpredictable and often mysterious results. This warning serves to alert the user that the presence of the **NAK** character(s) may be a problem.

Both warning messages can be suppressed by adding **WARN=NO** to the corresponding **#RJIN** command.

Warning: EOF on input file - *n* bytes remaining to be sent

When an **#RJIN** has been performed from $STDINX and end-of-file is encountered, this warning is displayed as a reminder that the file has not been completely transfered yet. Network/S must wait until the next command has been input before determining how to terminate the current file transfer.

Receiving *type* file data to file: *filename*

When routed data is received, this message is displayed to indicate the type of routing (List, Punch or un-routed) and the destination, if either the **LIST** or **PUNCH** options were specified in the **#RJOUT** command.

Warning: {RJIN | Command} file may be numbered

When Network/S opens either type of indicated file, it checks to see if the file was saved as a Numbered Editor file. If the file was indeed saved this way, Network/S will send the line numbering information in columns 73-80 as though they were data. This is almost certainly *not* what is intended. This test is only performed on fixed-length files, and only on the first record of each file. If the last eight columns of the first record are numeric, Network/S will display this warning.

Unexpected routed {punch | list} data set received

If, during the execution of either an **#RJLIST** or an **#RJPUNCH** command, routed punch or list data is received, respectively, this error results. If Network/S is running in a batch job, it will scan forward through the current command file for the next available, suitable, **#RJLIST** or **#RJPUNCH** (as the case may be) for the data just received.

Disc record size too large (*n*) - must be < 4096

An attempt was made to transmit a file whose record width (*n*) was greater than 4,096 bytes, the maximum that Network/S allows.

Length of last record read (*m*) is larger than the max buffer size (*n*)

During an **#RJIN** command, a record was read that is larger (*m* bytes) than the current maximum buffer size (*n* bytes), as dictated by the **#RJLINE** command. By default, in 3780 the maximum buffer size is 512 bytes and in 2780, it is 400 bytes. These maximums may be changed using the **MAXRPB** option, but may not exceed 4,096 bytes.

Warning: Disc record size (*m* bytes) is smaller than OUTSIZE (*n*) OUTSIZE ignored

If the **OUTSIZE** parameter has been specified in an **#RJOUT** command, and the **OUTSIZE** size (*n*) is greater than the record width of the indicated file (*m*), this warning will result.

Received record size too large - will {wrap | truncate} at *n* bytes

If, during the execution of an **#RJOUT** command, a logical record is received that is too wide for the output file, this message is displayed. If the **TRUNCATE** option was specified, the overflow will be discarded; otherwise, the overflow will be written to subsequent records.

Warning: empty data set skipped

If the **#RJIN** command is used to send an empty file, this message is displayed and the command is ignored.

Warning: Automatic RJOUT after RVI

If a Reverse-Interrupt (**RVI**) is received during the execution of an **#RJIN** command, and the **RVI** does not appear as the acknowledgment for the last block in the data set, Network/S will perform an **#RJOUT** $STDLIST automatically, prior to terminating.

Network Engine related errors

No response from Network Engine - Attempting recovery

Recoverable error - *error message*

If, at any point during Network/S' communications with the Network Engine, the Engine fails to respond to a read request, either due to a software time-out (the first message) or due to some other I/O error on the serial port (the second message) one of the messages is displayed and Network/S attempts to regain contact with the Engine. If the attempt is successful, the message:

Recovered

is displayed and the program continues normally. Network/S will attempt up to **RETRY** times to recover from these types of errors. Software time-outs, if they occur during installation, often indicate that the Network Engines DIP switches specify a speed different from that specified on the **#RJLINE** command. They can also indicate that the Engine is not turned on or not connected to the expected logical device.

If software time-outs occur after installation, check to see that the cables are tightly connected and that the Engine is still powered up. If trouble persists, contact Telamon.

Packet {length | format} error - Attempting Recovery

This message indicates a break-down in communications between the HP 3000 and the Network Engine. If it occurs more than once or twice, and is not recovered, contact Telamon.

Invalid *message* response (*text)*

This message indicates a break-down in communications between Network/S and the Network Engine. It identifies a unexpected response to an Engine command. If it recurs, contact Telamon.

Modem Call Progress Messages

Universal Data Systems: UDS201C/D, UDS208B/D or UDS2140 (UDS)

E	Error in dial message {fatal}
N	No dial tone detected {fatal}
D	Dial tone detected
R	Ring back tone detected
B	Busy line detected {fatal}
T	Re-order tone detected {fatal}
Q	Quit (Abort) timeout {fatal}
A	Call complete
I	Invalid answer tone detected {fatal}
?	Unknown response

Racal Vadic 4850PA (SADL)

D	Dialing
R	Ringing
A	Answer tone
L	Online
B	Busy {fatal}
F	Failed call {fatal}
E	No dial tone detected {fatal}
C	Invalid dial command {fatal}
?	Unknown response

Racal Milgo RMD3222, Racal Vadic RMD3222, Penril Alliance V.32, etc. (V.25 and V.25,ASCII)

VAL	Valid command
CNX	Connected
INV	Invalid command {fatal}
CFIET	Engaged tone (busy signal) {fatal}
CFICB	Modem busy {fatal}
CFIRT	Ring tone collision {fatal}
CFIAB	Call aborted {fatal}
CFINT	No dial tone {fatal}
CFIxx	(Any other Call Failure Indicator) {fatal}
???	Unknown response

801 Autodialers: UDS801A/C (801)

P	Power Failure {fatal}
N	Dialing (PND)
D	Modem in use (DLO) {fatal}
B	Busy/No answer (ACR) {fatal}
A	Quit (No COS/DSC) {fatal}
C	Answer (COS)
E	Dialing Error (ACR) {fatal}
???	Unknown response

APPENDIX F Programmatic Access

It is usually necessary to create command files that exactly specify the expected sequence of file transfers. For instance, if you expect to send one file to a remote system and then receive another file, you must first issue an **#RJIN**, followed by an #**RJOUT**, not the other way around. Consequently, the remote system should be set up to first receive a file and then send one back to you.

Programmatic access provides a means to partially ignore this sequencing restriction.

The original use of RJE was to perform job entry on a half-duplex communications link, where only one party to the conversation could be sending at a time. The IBM 2780/3780 devices, combination card reader/line printer stations, were used to submit batch jobs and to receive line printer output. In the steady state, the station was set to receive any output that might arrive from the host system. When someone wanted to submit a job, the station would have to be taken out of receive mode and placed into transmit mode in order to send the deck of cards containing the user's job. Once the deck had been sent, the station would automatically revert to receive mode, usually to receive the output of the job just submitted.

To accomplish this, Network/S (and HP's RJE) makes use of message files. A communications link is established normally, via the #**RJLINE** command, with the **MSGFILE** parameter specified. This parameter identifies a message file, normally located in **MSG.RJE**, that will receive requests for transmission from users/programs on the local system. Once an #**RJOUT** command has been issued with both the **REPEAT** and **INTERRUPT** options set to **YES**, the program will then await either an output data set from the remote host or an interrupt, via the **MSGFILE** file, requesting some other operation. If an output data set arrives, the **REPEAT=YES** option specifies that the original #**RJOUT** command will be re-executed upon completion. If an interrupt arrives, and no subsequent #**RJOUT** command explicitly disables **REPEAT** mode, the original **#RJOUT** command will be re-executed upon the completion of the interrupt command file.

For example, let's assume that an interrupt message file has been built in **MSG.RJE** as follows:

```
:BUILD REQUEST.MSG.RJE;MSG;REC=-36,,V,ASCII
```

The following command sequence will set up the default environment to receive output from a remote host and to service occasional requests from programs on the HP 3000:

```
#RJLINE 3780;MSGFILE=REQUEST
#RJCOMMENT Send initial ID sequence to remote host
#RJIN LOGIN
#RJCOMMENT Setup to receive multiple output sets
#RJOUT "LP";WAIT=0;REPEAT=YES;INTERRUPT=YES
```

The final #**RJOUT** command will execute indefinitely, until the line is dropped or until an interrupt file explicitly sets **REPEAT=NO**.

Now suppose that a user wishes to send a data file to the remote host, and expects no output. The user would create an unnumbered Editor-style command file containing:

```
#RJIN MYFILE;INCODE=ASCII;...
#RJEOD
```

and would name it, for this example, **MYCMD.PUB.MYACCT**. Now, to cause Network/S to interrupt the pending **#RJOUT** command to execute this command, the user could enter:

```
:FCOPY FROM=;TO=REQUEST.MSG.RJE;SUBSET=0,1
HP32212A.03.25 FILE COPIER (C) HEWLETT-PACKARD CO.  1984

MYCMD.PUB.MYACCT
1 RECORD PROCESSED *** 0 ERRORS

END OF PROGRAM
:
```

Network/S, if not in the midst of receiving a data set from the remote host, would temporarily suspend the **#RJOUT** command, execute **MYCMD.PUB.MYACCT**, and then re-execute the original **#RJOUT** command.

Now suppose that a second user wishes to send two data files to the remote system and then wait for one output file. The user would create a command file something like:

```
#RJIN FILE1;...
#RJIN FILE2;...
#RJEOD
#RJOUT FILE3
```

Note that this #**RJOUT** command does *not* specify **REPEAT** explicitly. This will leave the original #**RJOUT** command's intent in effect. If **REPEAT=NO** had been specified, since the user didn't want this command to repeat, the effect would have been to terminate the original command's execution, terminating Network/S also. Also note that **INTERRUPT** is not mentioned. This will prevent the user's #**RJOUT** command from being interrupted via the original **REQUEST** file specified in the #**RJLINE** command.

Once this file's name has been written to **REQUEST**, Network/S will execute the commands specified and will then re-execute the original, saved #**RJOUT** command upon completion.

To terminate a repeating command, an interrupt file could specify a command file containing either an **#RJEND** command or an **#RJOUT** command with **REPEAT=NO** specified explicitly.

APPENDIX G Sample Scripts

This appendix contains sample scripts for communicating with several Value-Added Networks (VANs). If you have been processing Electronic Data Interchange (EDI) transactions using programs provided by a third-party EDI software company, some changes may be required in these scripts. For example, many EDI software packages expect files to be in a certain format or they will be unable to process the data. Consult your EDI software company for instructions on sending and receiving files to your trading partners. Always rely on the documentation provided by the VAN for the latest, most accurate information. If you need to write a script for a VAN or a major company but are unclear on how to do so from the documentation provided by the VAN or company, call Telamon. We will be happy to assist you after receiving a copy of the documentation.

Some of these sample scripts have been submitted to us by our customers. Telamon does *not* maintain accounts with all VANs and these examples are meant to provide you with a starting point. See the *Getting Started* chapter for an example of how to build scripts.

Advantis (IBM Information Network) Sample Job

```
!JOB jobname,user/userpass.account/acctpass,group/grppass
!COMMENT +-----------------------------------------------------+
!COMMENT |This job logs on to the IBM Information Network (IIN)  |
!COMMENT |then logs on the Information Exchange interface.  The  |
!COMMENT |job first sends a file to an IIN mailbox.  The job then|
!COMMENT |receives a file from your IIN mailbox and writes it to |
!COMMENT |the HP 3000.                                           |
!COMMENT +-----------------------------------------------------+
!COMMENT Step 1 - Build an MPE file to write data to. If the build
!COMMENT          fails, check to see if the file exists. If the
!COMMENT          file does NOT exist then abort (because the
!COMMENT          build failed for some unexpected reason.) If the
!COMMENT          file exists, then erase it.
!COMMENT
!SETJCW JCW 0
!SETJCW CIERROR 0
!CONTINUE
!BUILD EDIRECV;REC=-80,16,F,ASCII;DISC=1000
!IF CIERROR <> 0 THEN
!   SETJCW CIERROR 0
!   CONTINUE
!   LISTF EDIRECV;$NULL
!   IF CIERROR = 0 THEN
!      COMMENT Erase the existing file
!      FILE ERASEME=EDIRECV;SAVE
!      PURGE *ERASEME
!      RESET ERASEME
!   ELSE
!      TELLOP +-------------------------------------------------+
!      TELLOP |IBM EDI Job Failed Because of File Build Error!!! |
!      TELLOP +-------------------------------------------------+
!      ABORT
!   ENDIF
!ENDIF
!COMMENT Step 2 - Run the Telamon Network program that communi-
!COMMENT          cates with IIN
!NERJE
#RJLINE 3780;LINECODE=EBCDIC;CONNECT=DIAL,"local IIN phone no.",modem-type
#RJBID
#RJOUT $STDLIST
#RJIN *;COMPRESS=NO;TRUNCATE=NO
/*LOGON iinacct,iinuserid,iinpass/*SELECT EDIRECT/*USERDATA BSCEDI
#RJEOD
#RJOUT $STDLIST
#RJIN *;COMPRESS=NO;TRUNCATE=NO
IELOGON ACCOUNT(iinacct) USERID(iinuserid) PASSWORD(iinpass);
#RJEOD
#RJOUT $STDLIST
#RJIN *;COMPRESS=NO;TRUNCATE=YES
SENDEDI;
#RJIN SENDDATA;COMPRESS=NO;TRUNCATE=NO
```

```
#RJEOD
#RJOUT $STDLIST
#RJIN *;COMPRESS=NO;TRUNCATE=NO
RECEIVE CLASS(X12);
#RJEOD
#RJOUT $STDLIST
#RJCOMMENT The EMPTYOK option below will handle the case when there
#RJCOMMENT   is no mail and your profile doesn't specify that a blank
#RJCOMMENT   line is to be sent.
#RJOUT EDIRECV;NL=NO;EMPTYOK
#RJCOMMENT This next step requests a log of the transactions with the
#RJCOMMENT   IIN. When the IIN receives the RECEIVE CLASS(EDILOG)
#RJCOMMENT   command, it sends three files back; a file acknowledging
#RJCOMMENT   the request for the log, the log itself and a logoff
#RJCOMMENT   message. If you do NOT request a log, you can log off
#RJCOMMENT   from the IIN by sending the following command sequence:
#RJCOMMENT   #RJIN *;COMPRESS=NO;TRUNCATE=NO
#RJCOMMENT   /*LOGOFF
#RJCOMMENT   #RJEOD
#RJCOMMENT   #RJOUT $STDLIST
#RJCOMMENT   #RJOUT $STDLIST
#RJCOMMENT   There are two #RJOUT commands above; one to receive the
#RJCOMMENT   acknowledgment of the logoff request and one to receive
#RJCOMMENT   the logoff text.
#RJIN *;COMPRESS=NO;TRUNCATE=NO
RECEIVE CLASS(EDILOG);
#RJEOD
#RJOUT $STDLIST
#RJOUT $STDLIST
#RJOUT $STDLIST
#RJEND
!TELLOP +-----------------------------------------------------+
!TELLOP |IBM EDI Job Finished Successfully.                    |
!TELLOP +-----------------------------------------------------+
!EOJ
```

GE Information Services (GEIS) Sample Scripts

Three GEIS scripts are presented here. The first script is a test script for logging into GEIS. The second script is an example of how to send data to GEIS. The third script is an example of how to receive data from GEIS.

GEIS presents a small challenge, due to its bisynchronous protocol implementation. At the end of a session, when it would otherwise be expected to send an **EOT** followed by a **DLE-EOT**, GEIS sends a **DLE-EOT** only. **DLE-EOT** is normally sent just before a line disconnect, while an **EOT** is normally sent at the end of a transmitted file. GEIS, and others, sometimes skip the **EOT** if it is to be immediately followed by a **DLE-EOT**. The problem is that Network/S normally treats an unexpected **DLE-EOT** as an error. To get around this "feature", the **XEND** parameter is used on the **#RJLINE** command so that the **DLE-EOT** sequence that GEIS sends is treated as an **EOT**.

If you are using the "newline character" (EBCDIC hex 15, ASCII hex 85) as a segment terminator, then the **NL=NO** parameter should be used on either the **#RJLINE** or the **#RJOUT** command. See the **NL=NO** parameter in the **#RJLINE** command in the *Commands* chapter for more details.

Note that when using the **3780** protocol, you must place a "*" in column 80 of the first transmitted record, the sign-on card. This flag indicates your intention to use **3780**.

Finally, note that your user-profile at GEIS may affect the way that GEIS sends data back to you.

GEIS Check out script

```
#RJLINE 3780;LINECODE=EBCDIC;TABLE=AEIBM.NETWORKS.TELAMON; &
#  CONNECT=DIAL,"GEIS phone number",modem-type;XEND
#RJIN *
ANS09102,COMMTEST,MAILA                                          *  {column 80}
*LTID MAILBOXA
*MODE INPUT,WAIT
PRINT MESSAGE;MAILBOXA;NONE
*EOS
#RJEOD
#RJSYS FILE LP;DEV=LP,1;CCTL
#RJOUT *LP
#RJSYS RESET LP
#RJEND
```

GEIS Send Example

The following is a Network Script for sending data to GEIS. Along with the appropriate GEIS job control languate, it will send the contents of the file ***EDISEND***.

```
#RJLINE 3780;LINECODE=EBCDIC;TABLE=AEIBM.NETWORKS.TELAMON; &
#  CONNECT=DIAL,"GEIS phone number",modem-type;XEND
#RJIN *
ABCnnnnn,PASSWD,MAILA                                            *  {column 80}
*LTID MAILBOXA
*MODE INPUT(OUTPUT(HISTnnn),LIST),WAIT,TAB(HSSTABLE)
*DATA DOCSID(PURE,ASCII)
```

```
#RJIN EDISEND
#RJIN *
*EOF
/EDXSND DOCSID
*EOS
#RJEOD
#RJOUT $STDLIST
#RJEND
```

GEIS Receive Example

The following are MPE commands and a Network Script for receiving data from GEIS. GEIS routes data to both list and punch. List data consists of "Welcome to GE" and status messages. Punch data is actual data (i.e. X12 EDI data). Running Network/S with parameters to specify list and punch files will insure the data is routed to the correct destinations.

Use the following commands (substitute the filename of your command file for ***CMDFILE***):

```
:PURGE GEDATA
:BUILD GEDATA;REC=-512,,V,ASCII
:FILE LP;DEV=LP,1;CCTL
:NERJE CMDFILE,,*LP,GEDATA
:RESET LP
```

The *nnnnn* in the sign-on record should be your valid GEIS user ID, and the *nnn* in the **MODE** record should be the last 3 numbers of your GEIS user ID.

Routed list data will be written to ****LP*** and routed punch data will be written to ***GEDATA***.

CMDFILE contains:

```
#RJLINE 3780;LINECODE=EBCDIC;TABLE=AEIBM.NETWORKS.TELAMON; &
#  CONNECT=DIAL,"GEIS phone number",modem-type;XEND
#RJIN *
ABCnnnnn,PASSWD,MAILA                                          *  {column 80}
*LTID MAILBOXA,CPUNCH,BUFFER(512)
*MODE INPUT(OUTPUT(HISTnnn),LIST),WAIT,TAB(HSSTABLE)
/EDXRCV PRIOR
*EOS
#RJEOD
#RJOUT
#RJEND
```

Ordernet Sample Script

```
#RJCOMMENT If the initial '$$' lines are sent in transparent mode, Ordernet
#RJCOMMENT   will send data back to you transparently. Use the #RJIN XPARENT
#RJCOMMENT   option if your data requires it.
#RJLINE 3780;LINECODE=EBCDIC;CONNECT=DIAL,"Ordernet phone number",modem-type
#RJIN *
$$REQUEST ID=userid BATCHID="password"
$$ADD ID=userid BATCHID="password"
#RJIN SENDDATA
#RJEOD
#RJOUT RECVDATA;NL=NO
#RJEND
```

Ordernet generally requires that the communications process be reduced to two parts: input to Ordernet and output from Ordernet. This requires us to combine all input requests in a sequence of consecutive **#RJIN** commands, followed by a single **#RJOUT** command to receive the response(s). Ordernet will only process these requests once both **#RJIN**s have completed. At that point, Ordernet will process the **$$REQUEST** command, resulting in data being sent back to you to be received into *RECVDATA*, followed by the **$$ADD** command, which will consume the contents of *SENDDATA* that follows the **$$ADD** command in the input data stream.

When a receive request is combined with a send request, Ordernet requires that the receive be processed first, followed by the send request. This refers to the order of the "**$$**" commands, *not* the order of Network/S commands. The Network/S commands *always* appear in the above order, **#RJIN**(s) followed by an optional **#RJOUT**. If you only expect to send data to Ordernet, omit the **$$REQUEST** line and the **#RJOUT** command. If you only expect to receive from Ordernet, omit the **$$ADD** line and the second **#RJIN**.

MCI Sample Script

The MCI network is unusual in that it requires that the logon information be sent in an **ETX**-terminated data block. The **NOETB** keyword on the first **#RJIN** command specifies this capability.

```
#RJLINE 3780;CONNECT=DIAL,"MCI phone number",modem-type;XEND;LINECODE=EBCDIC
#RJIN *;TRUNCATE=NO;COMPRESS=YES;NOETB
TELAMNEDI 1111111;
#RJIN SENDDATA
#RJEOD
#RJOUT RECVDATA
#RJEND
```

APPENDIX H NETTRANS

NETTRANS - 3270/3780 Transaction Processor

NETTRANS is a special purpose application, designed to be used in an online, transaction environment, where the need exists to emulate a communications controller operating in real-time. With **NETTRANS**, the Network Engine can be setup to emulate:

- an IBM 3274 Cluster controller, responding to polls from a mainframe controller, such as an IBM 3705 or equivalent. (**3270 MASTER=NO**)
- an IBM 3705 controller (or equivalent), polling one or more IBM 3274 Cluster controllers. (**3270 MASTER=YES**)
- an IBM controller employing the point-to-point protocol used by IBM 3780 devices. (**3780**)

Typical applications include real-time credit card validation services and communicating with bank Automated Teller Machines (ATMs). If you have a specific need for these kinds of capabilities, read on. **NETTRANS** doesn't have a user interface, per se, and isn't usable except in conjunction with a user-written application.

NETTRANS routes transactions between a serial RS-232 port and the Synchronous Network Engine and communicates with a user application via a pair of message files:

```
NTINPUT     - Transaction/command input
NTOUTPUT    - Transaction/message output
```

When **NETTRANS** is run, it will build these two files if they are not present - old files will be used if they are found. **NTOUTPUT** is opened for append access, so any data in it will be preserved.

Command Line

```
:RUN NETTRANS.NETWORKS.TELAMON;INFO="TXLINE command"
```

Installation

NETTRANS is stored on a magnetic tape containing the program and associated files. To install the software, mount the tape on your tape drive and type the following commands:

```
:HELLO MANAGER.SYS
:FILE TELAMON;DEV={TAPE DEVICE}
:RESTORE *TELAMON;TELAMON.PUB.SYS;SHOW
```

When the **TELAMON** file has been restored, type:

```
:RUN TELAMON.PUB
```

This program will prompt you for various user and account passwords and the *tape device* specified in the **:FILE** equation above. The **MANAGER.SYS** passwords are your existing passwords and the **MANAGER.TELAMON** passwords will become the new passwords for **MANAGER.TELAMON**. The program will prompt you to assign them. The program will create the job streams necessary to build and/or re-configure the **TELAMON** account and to restore the files from your installation tape.

When the stream file has been created, the program will prompt you again to confirm whether you want to stream the job. Enter Y to stream the job and N to suppress the stream. If you enter Y, the job will be streamed and the stream file will be purged. If you enter N, you will be prompted to save the created job stream. If you enter Y, the job stream will be saved as **$OLDPASS** allowing you to view or manually stream the job. If you enter N, the job stream will be purged.

Operation

If ***TXLINE command*** is specified in the **INFO** string, it will be interpreted as described below.

NETTRANS reads both commands and transactions from **NTINPUT**, with the following formats:

```
         1         2
12345678901234567890...
CD ... Data ...
UA
```

for 3270 data and

```
         1         2
12345678901234567890...
ss... Data ...
pp
```

for 3780 data, and

```
         1
123456789012345...
#command
```

for commands.

If the first character of the record is a pound sign (#), **NETTRANS** will attempt to interpret the remaining portion of the record as a command (see below.) Otherwise, the first two characters are treated as either valid CU|DA characters (3270) or blanks (3780). For 3270 mode, the CU character will be validated against the table shown in the description of the **POLL** keyword, below. For 3780 mode, the two blank characters (**sp**) will be deleted before the data is sent to the Engine.

Both received transactions and messages from **NETTRANS** will be written to **NTOUTPUT**, with messages being prefaced with a pound sign (#) and a four digit message number. (See the Error Messages section, below.) Received transactions will be prefaced with either a CU|DA address sequence (3270) or two blanks (3780).

Additionally, commands read from **NTINPUT** will be "echoed" to **NTOUTPUT**.

#TXLINE 3270 or 3780

The Synchronous Network Engine is capable of operating in the **3270** multi-point and the **3780** point-to-point processing modes.

Syntax:

```
#TXLINE 3270;POLL=Control Unit IDs
```

or

```
#TXLINE 3780
```

Default:

None - this parameter must be specified explicitly

#TXLINE LINECODE

The **LINECODE** parameter is used to specify the line transmission code to be used. The possible values are **ASCII** and **EBCDIC** and must match the codes used by the remote host.

By default, when **LINECODE=EBCDIC** is selected, a Cyclic Redundancy Check (**CRC**) checksum scheme is employed during data transfer. When **ASCII** is selected, a Longitudinal Redundancy Check (**LRC**) scheme is used for non-transparent transmissions; for transparent transmissions **ASCII,CRC** is used automatically. To override these defaults, append either **CRC** or **LRC**, as is appropriate, to the transmission code.

Syntax:

```
#TXLINE ... ;LINECODE={EBCDIC | ASCII} [,{CRC | LRC}]
```

Default:

```
#TXLINE ... ;LINECODE=EBCDIC,CRC
```

#TXLINE TXCODE

The **TXCODE** parameter is used to identify the expected character set for the data to be sent by this **#TXLINE** command. The Engine uses this information the translate the data before it is sent. The translation is performed according to the **#TXLINE ... ;LINECODE** setting and is done in such a way that all data sent is of the same language. The possible values are:

ASCII	The data is ASCII and is converted if **LINECODE=EBCDIC**
EBCDIC	The data is EBCDIC and is converted if **LINECODE=ASCII**
BINARY	The data is not to be converted during transmission (**XPARENT=YES** is recommended)

Syntax:

```
#TXLINE ... ;TXCODE={ASCII | EBCDIC | BINARY}
```

Default:

```
#TXLINE ... ;TXCODE=ASCII
```

#TXLINE XPARENT

The **XPARENT** parameter is used to inform the Engine that the data that will be sent may contain characters that can be misinterpreted as bi-synchronous control characters. When set to **YES**, the Engine will 'protect' certain characters before sending them by inserting a Data-Link-Escape (**dle**) character in front of each.

Syntax:

```
#TXLINE ... ;XPARENT={YES | NO}
```

Default:

```
#TXLINE ... ;XPARENT=NO
```

#TXLINE CONNECT

The **CONNECT** keyword, if specified, indicates that a switched-line, dialup connection is to be attempted, either originating from the local system or from the remote system.

If the **CONNECT** keyword is omitted, a direct-connect link is assumed.

#TXLINE CONNECT=DIAL

To specify a connection originating from the local system, enter:

```
#TXLINE ... ;CONNECT=DIAL[,"phone number"[,modem type]]
```

The *phone number* should be the exact string of text to be used by the modem, or by the operator should this be a manual dial attempt.

modem type, if specified, should be one of the following values:

DTR — Manual dial (default). The Engine raise the Data-Terminal-Ready (**DTR**) signal to modem and will then wait for a connection occur. With this setting, a message will appear on std-out requesting that the specified phone number be dialed.

UDS — Auto-dial using a Universal Data Systems modem, either a UDS201C/D or a UDS208B/D. These modems can be automatically dialed from the Engine. The specified phone number will be passed to the modem as-is. The UDS modems require a dial command in the form:

Dphone info

Note that the Engine will supply the 'D' - the remainder of the information will be taken from the specified phone number.

SADL — Auto-dial using a modem that supports the Synchronous Auto Dial Language dialing method, for instance Racal-Vadic model 4850PA. These modems can be automatically dialed from the Engine. The specified phone number will be passed to the modem as-is.

V.25 — Auto-dial using any V.25bis compatible synchronous auto-dial modem, such as the Codex 2234/2264 or the UDS2140. Optionally append ",**ASCII**" to specify that the V.25 dialing take place using ASCII character codes. The default is EBCDIC.

801 — Auto-dial using the Bell 801 protocol. This option requires a special version of the Synchronous Network Engine that supports the RS-366 parallel interface to the auto-dial unit.

phone number must be specified if manual dialing is not to be used.

Syntax:

```
#TXLINE ... ;CONNECT=DIAL,"phone"[,DTR|SADL|UDS|V.25[,ASCII]|801]
```

Example:

```
#TXLINE ... ;CONNECT=DIAL,"5551234",SADL
```

#TXLINE CONNECT=ANSWER

To specify a connection originating from the remote system, enter:

```
CONNECT=ANSWER
```

The Engine will wait for either an incoming call to arrive or until the time limit, specified by the **WAIT** parameter, expires.

Note that the modem must provide the Ring Indicator (**RI**) signal on pin 22 of the RS-232 connection to the Engine - the Engine won't raise **DTR** to the modem until **RI** switches from Off to On. If your modem doesn't support **RI**, omit the **CONNECT** sequence from the **#TXLINE** command - the Engine will correctly answer the incoming call without waiting for the **RI** signal.

Syntax:

```
#TXLINE ... ;CONNECT=ANSWER
```

#TXLINE DEV

The **DEV** keyword is used to identify the logical device to which the Synchronous Network Engine is attached and to specify the speed to be used between the Engine and the local computer's terminal port. The format is:

```
DEV=[device name/number] [,speed]
```

If the *speed* parameter is omitted, a value of 960 cps will be used.

Note that the *speed* setting must match the DIP-switch settings on the Engine, positions 1-4. If these values do not match, **NETTRANS** won't be able to communicate with the Engine!

Syntax:

```
#TXLINE ... ;DEV=[device name/number] [,speed]
```

Default:

```
#TXLINE ... ;DEV=RJLINE,960
```

Note that the *device name* and *speed* may be specified in the **RJECLINE** file.

#TXLINE WAIT

If either **CONNECT=ANSWER**, **CONNECT=DIAL,"...",DTR** or a direct-connect link is being attempted, the **WAIT** parameter will specify how long **NETTRANS** will wait for the connection to be made. The value is specified as follows:

```
WAIT=seconds
```

For example, to wait a maximum of 5 minutes for an incoming call, you would enter:

```
#TXLINE ... ;CONNECT=ANSWER;WAIT=300
```

Note that if **DIAL**,...,**UDS**, **SADL**, **V.25** or **801** is specified, the modems will dictate the wait period. Most modems wait about 90 seconds for the answering modem to go off hook before aborting the attempt.

Specifying **WAIT**=0 will cause the program to wait indefinitely.

Syntax:

```
#TXLINE ... ;WAIT=seconds
```

Default:

```
#TXLINE ... ;CONNECT=ANSWER;WAIT=0
#TXLINE ... ;WAIT=180
```

#TXLINE MASTER

The **MASTER** parameter is used to select primary or secondary contention in **3780** mode. In **3270** mode, this parameter specifies whether the Engine is the Master (polling device) or the Slave (polled device).

Syntax:

```
#TXLINE ... ;MASTER={YES | NO}
```

Default:

```
#TXLINE ... ;MASTER=NO
```

#TXLINE 3270 POLL

The **POLL** parameter is used to describe to the Engine the valid polling addresses to send and receive and is only required for **3270** operation. A list of values can be specified, separated by commas, with the values ranging from 0 to 30. These values correspond to the following tables:

3270 CU (Poll)/DA Characters

```
Index:      00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
Graphic:    SP A  B  C  D  E  F  G  H  I  [  .  <  (  +  !
ASCII Hex:  20 41 42 43 44 45 46 47 48 49 5B 2E 3C 28 2B 21
EBCDIC Hex: 40 C1 C2 C3 C4 C5 C6 C7 C8 C9 4A 4B 4C 4D 4E 4F

Index:      16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Graphic:    &  J  K  L  M  N  O  P  Q  R  ]  $  *  )  ;  ^
ASCII Hex:  26 4A 4B 4C 4D 4E 4F 50 51 52 5D 24 2A 29 3B 5E
EBCDIC Hex: 50 D1 D2 D3 D4 D5 D6 D7 D8 D9 5A 5B 5C 5D 5E 5
```

3270 CU (Select) Characters

```
Index:      00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
Graphic:    -  /  S  T  U  V  W  X  Y  Z  |  ,  %  _  >  ?
ASCII Hex:  2D 2F 53 54 55 56 57 58 59 5A 7C 2C 25 5F 3E 3F
EBCDIC Hex: 60 61 E2 E3 E4 E5 E6 E7 E8 E9 6A 6B 6C 6D 6E 6F

Index:      16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Graphic:    0  1  2  3  4  5  6  7  8  9  :  #  @  ‘  =  “
ASCII Hex:  30 31 32 33 34 35 36 37 38 39 3A 23 40 27 3D 22
EBCDIC Hex: F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 7A 7B 7C 7D 7E 7F
```

The first two characters of all **3270** transaction records written to the socket must be found in the appropriate table above, in the ASCII Hex rows. ASCII-to-EBCDIC conversion will be performed by the Network Engine. **NETTRANS** verifies that the index of the first character, the CU, was specified in the poll list and that the DA character is found on the list.

Syntax:

`#TXLINE 3270 ... ;POLL=`*`idx0`*`[,`*`idx1`*`...]`

Default:

None - this parameter must be specified explicitly in 3270 mode

#TXLINE RELIABLE

The **RELIABLE** parameter is used to enable/disable reliable mode. When enabled, the user application is responsible for providing two additional bytes in each outbound data record. These two ID bytes should be placed immediately in front of the data portion of the transaction, skipping any CU and DA or spaces in **3780** . The Engine will remove these bytes before the transaction is sent. When, and if, the transaction is positively acknowledged (**ack**) or negatively acknowledged (**nak**), a message containing the ID bytes will be generated and sent back to the user application. See the discussion on **Reliable Mode** that follows.

Syntax:

```
#TXLINE ... ;RELIABLE={YES | NO}
```

Default:

```
#TXLINE ... ;RELIABLE=NO
```

#TXLINE 3270 LIMIT

If **MASTER=YES** and the protocol is **3270**, **LIMIT** specifies the maximum number of times a Control Unit can fail to respond before being placed on the Slow Poll list. It can range from 0 to 63, with 0 implying no limit. See **SLOWPOLL**, below.

Syntax:

```
#TXLINE 3270 ... ;LIMIT=timeout limit
```

Default:

```
#TXLINE 3270 ... ;LIMIT=0
```

#TXLINE 3270 SLOWPOLL

If **MASTER=YES** and the protocol is **3270**, **SLOWPOLL** specifies the number of polls skipped for CUs on the Slow Poll list. It can range from 0 to 63 with zero implying no polling for devices on the list.

Syntax:

```
#TXLINE 3270 ... ;SLOWPOLL=n
```

Default:

```
#TXLINE 3270 ... ;SLOWPOLL=0
```

#TXLINE MAXTRAN

If the protocol is **3780**, this parameter specifies the maximum number of transactions that will be sent during any single bid sequence. It can range from 0 to 15. If set to 1 or more, the Network Engine will send no more than **MAXTRAN** transactions after each bid, after which an **eot** will be sent to return the line to the control state. If fewer than **MAXTRAN** transactions are in the Engine's transmit buffer, the Engine will send an **eot** as soon as the buffer is emptied. If set to 0 (zero), the client application is responsible for the **eot** sequence. If the client application fails to send data to the Engine in a timely manner, the Engine will automatically send **ttd**s.

Syntax:

```
#TXLINE 3780 ... ;MAXTRAN=n
```

Default:

```
#TXLINE 3780 ... ;MAXTRAN=4
```

#TXLINE 3270 IDLE

If **MASTER=NO** and the protocol is **3270**, **IDLE** specifies a timer that the Engine will employ to measure time between Polls, or Selects, from the Master system. Should the specified amount of time expire with no activity, the Engine will send a Poll Idle message to **NETTRANS** and message **0027** will be written to **NTOUTPUT**. **IDLE** is specified in centi-seconds.

Syntax:

```
#TXLINE 3270 ... ;IDLE=centi-seconds
```

Default:

```
#TXLINE 3270 ... ;IDLE=0
```

#TXLINE BAUDRATE

The **BAUDRATE** keyword specifies the speed that the Engine is to use either for its internal synchronous clock. The acceptable values are:

```
1200, 2400, 4800, 7200, 9600, 19200
```

Syntax:

```
#TXLINE ... ;BAUDRATE=baud rate
```

Default:

```
#TXLINE ... ;BAUDRATE=4800
```

#TXLINE ID

The **ID** keyword specifies the local terminal identification string used by the Engine when a communications link is first made. Use this feature only if required by the remote system.

The **ID** string is interpreted as an ASCII string. If the required identification sequence is a printable sequence of characters, any necessary translation will be performed by the Engine - e.g. ASCII to EBCDIC. If the necessary id sequence is not a printable string, you will need to load this sequence into a text file, to be referenced using the

```
... ;ID="^FILENAME" ...
```

.format.

If the **LINECODE** is **EBCDIC**, and the **ID** string is non-printing, you will need to first translate the sequence from **EBCDIC** to **ASCII** before placing this string in the file. The Engine will then translate the data back to **EBCDIC** before sending it on to the host system.

Syntax:

```
#TXLINE ... ;ID="local id"
```

or

```
#TXLINE ... ;ID="^filename"
```

Default:

None

#TXLINE REMID

The **REMID** keyword specifies the remote terminal identification string to be recognized by the Engine when a communications link is first made. Specify this string only if you expect the remote system to send its local ID string during a line bid sequence.

The **REMID** string is interpreted as an ASCII string. If the required identification sequence is a printable sequence of characters, any necessary translation will be performed by the Engine - e.g. **ASCII** to **EBCDIC**. If the necessary **REMID** sequence is not a printable string, you will need to load this sequence into a text file, to be referenced using the

```
... ;REMID="^FILENAME" ...
```

format.

If the **LINECODE** is **EBCDIC**, and the **REMID** string is non-printing, you will need to first translate the sequence from **EBCDIC** to **ASCII** before placing this string in the file. The Engine will then translate the data back to **EBCDIC** before comparing it to the received identification sequence.

Syntax:

```
#TXLINE ... ;REMID="remote id"
```

or

```
#TXLINE ... ;REMID="^filename"
```

Default:

None

#TXLINE TABLE

The Network Engine performs any and all necessary ASCII-to-EBCDIC and EBCDIC-to-ASCII translations during file transfers. The default translation table used by the Engine was derived from that used by Hewlett-Packard's RJE product. This is also the table supported by the MPE **CTRANSLATE** intrinsic.

The *IBM 3780 Component Description* manual specifies a slightly different set of translations. The majority of the printable characters are translated the same as with HP's table, but there are a few characters that have different values. In particular, the two tables specify different translations for the ASCII exclamation point character (“!”). HP's translation converts this character to an EBCDIC value of 79 decimal, or 4F hex. IBM's translation converts to 90 decimal, or 5A hex. There are other differences, of course, but this character is the most noticeable. Both tables are listed in Appendix B, *Character Sets.*

The Engine can be instructed to use an alternate translation table - the **TABLE** option is used to specify this table. The translation information is located in a specially formatted file, provided by Telamon. Two such files are found in the **NETWORKS** group of the **TELAMON** account:

```
AEHP.NETWORKS.TELAMON
```

and:

```
AEIBM.NETWORKS.TELAMON
```

These two files appear as tables appear in Appendix B. The **AEHP** table is the default table used by the Engine - it need not be specified as its values are permanently loaded in the Engine's firmware. The **AEIBM** table contains the values matching those found in the *IBM 3780* manuals. To specify this table, enter:

```
#RJLINE ... ;TABLE=AEIBM.NETWORKS.TELAMON
```

or:

```
TABLE = AEIBM.NETWORKS.TELAMON
```

in the **RJECLINE** file.

Don't attempt to modify these files. NETTRANS performs a number of internal consistency checks on the data prior to downloading the table to the Engine - if the format isn't valid, the program will terminate with an error condition. If different translations are required, contact Telamon and we will try to create a suitable translation table for you.

Syntax:

```
#TXLINE ... ;TABLE={filename}
```

Default:

Default HP ASCII-to-EBCDIC table is used.

#TXLINE ADDEOL

Normally, **NETTRANS** removes all bisync control characters received from the remote system before writing data to **NTOUTPUT**. If **ADDEOL** is set to **YES**, the terminating **etx**/**etb** characters will be included in the data record.

Syntax:

```
#TXLINE ... ;ADDEOL={YES | NO}
```

Default:

```
#TXLINE ... ;ADDEOL=NO
```

#TXLINE USERFRAME

Similar in function to **ADDEOL**, **USERFRAME** specifies that both the starting and ending framing characters will be included in **NTOUTPUT** and **NTINPUT**. **USERFRAME** additionally requires that data records written to **NTINPUT** have the framing characters specified. Normally, **NETTRANS** will provide **stx** and **etx** by default.

Due to the constraints of maintaining backwards compatibility with previous versions of **NETTRANS**, there are two forms of User Framing. The first, with **USERFRAME** set to **YES**, specifies a format where the start character, (**stx** or **soh**) is to be found in the *third* character position in the record, immediately following the CU|DA in **3270** mode or the two blank characters in **3780** mode. The second format, with **USERFRAME** set to **UNIX** (so named as to make the record format match that used by the Unix implementation of **NETTRANS**) specifies that the start character is to be found in the *first* character position, preceding any data. Additionally, in **3780** mode, the **UNIX** form does not require the two blank characters preceding the data. With either form, the *last* character of each record must be either **etx** or **etb**.

These formats apply to user data written to **NTINPUT** as well.

For example, with **USERFRAME** set to **YES**:

```
         1        2
12345678901234567890...
CDs ... Data ... e
```

```
UAt              t
  x              x
```

for **3270** data and:

```
         1         2
12345678901234567890...
sss... Data ...e
ppt             t
  x             x
```

for **3780** data, and with **USERFRAME** set to **UNIX**:.

```
         1         2
12345678901234567890...
sCD ... Data ... e
tUA             t
x               x
```

for **3270** data and:

```
         1         2
12345678901234567890...
s... Data ...e
t             t
x             x
```

Syntax:

```
#TXLINE ... ;USERFRAME={YES | NO | UNIX}
```

Default:

```
#TXLINE ... ;USERFRAME=NO
```

#TXLINE ROUTER

This option will cause the specified run command to be executed by **NETTRANS** as soon as the **NTINPUT** and **NTOUTPUT** files have been created. run command can be any valid MPE-style :RUN command, minus the ":RUN " portion. e.g.

```
ROUTER='MYPROG.TEST;LIB=G'
```

This process can be used to route messages and data from **NETTRANS** to appropriate locations.

Syntax:

```
#TXLINE ... ;ROUTER="run command"
```

Default:

None

#TXLINE STATUS

If **STATUS** is set to **YES**, **NETTRANS** will modify all messages sent to **NTOUTPUT** to include a flag indicating whether or not the Engine was online when the message was generated. This flag, located immediately after the "#nnnn" portion of the message, will be a "+" if the Engine reported online and a "-" if offline.

The flag will take the place of the blank character that normally separates the message number from the message text.

Syntax:

```
#TXLINE ... ;STATUS={YES | NO}
```

Default:

```
#TXLINE ... ;STATUS=NO
```

#TXLINE LOAD

NETTRANS, by default, writes all messages and transactions to the **NTOUTPUT** message file. The **LOAD** option can be used to specify an external procedure that can, in turn, be used to process the data prior to its being written to the message file, or even to suppress its being written at all. The procedure is specified using:

```
LOAD=procedurename [ ( lib ) ]
```

NETTRANS will attempt to load the specified procedure, in the exact specified case, from the SL or XL file specified by *lib.*

When running **NETTRANS** on an MPE/V based HP 3000 or in Compatibility Mode on an MPE/XL or MPE/iX based HP 3000, *Lib* can be one of the following:

Lib	SL Search sequence	
---	---	
G	SL.*logongroup*.*logonaccount*	then
	SL.PUB.*logonaccount*	then
	SL.PUB.SYS	
P	SL.PUB.*logonaccount*	then
	SL.PUB.SYS	
S	SL.PUB.SYS only	
GX	SL.*appgroup*.*appaccount*	then
	SL.PUB.*appaccount*	then

	SL.PUB.SYS	
PX	SL.PUB.*appaccount*	then
	SL.PUB.SYS	

where *logongroup* and *logonaccount* refer to the user's logon group and account and *appgroup* and *appaccount* refer to the location where the **NETTRANS** application resides, by default, NETWORKS and TELAMON respectively.

When running the Native Mode version of **NETTRANS** on an MPE/iX based HP 3000, **NETTRANS** will attempt to load the specified procedure from the XL file identified by *lib*, which must be a valid filename.

In SPL/V notation, the procedure is declared as follows:

```
integer procedure PROC'NAME(L'BUFFER, BUFFER'LEN);
   logical array           L'BUFFER;
   integer                          BUFFER'LEN;
```

In C notation, the procedure header would be:

```
short int PROC_NAME(BUFFER, BUFFER_LEN)
   char
      *BUFFER;
   short int
      *BUFFER_LEN;
```

In PASCAL notation, the procedure header would be:

```
procedure PROC_NAME(
      var BUFFER : packed array [0..4095] of char;
      var BUFLEN : shortint;
               ) : shortint;
```

NETTRANS will call this procedure, passing the transaction in **L'BUFFER** and the size of the transaction, in bytes, in **BUFFER'LEN**. Upon return from the procedure, **NETTRANS** will continue processing the data if **BUFFER'LEN** contains a positive, non-zero value.

The integer value returned by the procedure can be used to identify a different file for **NETTRANS** to use for output. By use of the **FILES** option, below, **NETTRANS** can be directed to open a number of output message files. If the library procedure returns a value of zero (0), **NETTRANS** will write the data to **NTOUTPUT**. If a positive value (*n*) is returned, and that value is less than or equal to the **FILES** count, the transaction will be written to the *n*th file. If the procedure returns a **-1**, **NETTRANS** will terminate. Any other return value will be ignored

Syntax:

```
#TXLINE ... ;LOAD=procname(lib)
```

Default:

None

#TXLINE FILES

For use with the **LOAD** option, above. A positive, non-zero value will cause **NETTRANS** to open a number of additional output message files, named **NTOUTP*nn***, where ***nn*** ranges from **1** to the value specified by **FILES**. These files will be used, if **NETTRANS** is so instructed, after a call to the **LOAD** procedure, as described above.

Syntax:

```
#TXLINE ... ;FILES=numfiles
```

Default:

```
#TXLINE ... ;FILES=0
```

#TXLINE CRASH

Specifies the Crash Timer, in seconds. This timer is used to cause the Network Engine to reset itself in the event that the **NETTRANS** program and/or the Local System fails to communicate with it. If the designated time interval passes and the Engine receives no data or commands from the **NETTRANS** process, a reset will occur and any open connection will be terminated.

Syntax:

```
#TXLINE ... ;CRASH=crash timer
```

Default:

```
#TXLINE ... ;CRASH=600
```

#TXLINE SYN

Specifies the number of **syn** (**syn**chronous idle) characters that will be sent at the start of each packet of information sent by the Engine.

Syntax:

```
#TXLINE ... ;SYN=syn count
```

Default:

```
#TXLINE ... ;SYN=4
```

#TXLINE NAK

Specifies the maximum number of consecutive **nak**s (**n**egative **a**c**k**nowledge) that can be sent before failure. **nak**s are sent when checksum errors are detected in received data.

Syntax:

```
#TXLINE ... ;NAK=nak count
```

Default:

```
#TXLINE ... ;NAK=6
```

#TXLINE ENQ

Specifies the maximum number of consecutive **enq**s (**enq**uire) that can be sent before failure. **enq**s are sent to the start a **3780** data transfer and as a status check when the receiving system fails to respond to data sent.

Syntax:

```
#TXLINE ... ;ENQ=enq count
```

Default:

```
#TXLINE ... ;ENQ=6
```

#TXLINE TTD

Specifies the maximum number of consecutive **ttd**s (**t**emporary **t**ext **d**elays) that can be sent before failure. **ttd**s are sent when **NETTRANS** is in the process of sending data and there is presently no data in the Engine to be sent. This may occur if the Local System becomes overloaded and cannot send data to the Engine fast enough for the Engine to continue transmitting uninterrupted.

Syntax:

```
#TXLINE ... ;TTD=ttd count
```

Default:

```
#TXLINE ... ;TTD=6
```

#TXLINE DIALWAIT

Specifies the maximum amount of time (in 4-second intervals) that the Engine will wait for a response to a dial sequence. Increase this value if your modem takes an unusually long time to connect.

Syntax:

```
#TXLINE ... ;DIALWAIT=dial wait count
```

Default:

```
#TXLINE ... ;DIALWAIT=20
```

#TXLINE TRACE

The **TRACE** keyword enables the creation of a diagnostic dump file, **XFERDUMP**, for later analysis via the **FMTDUMP** program, accessed using the **#FMTDUMP** command.

If **TRACE** is set to **YES**, **NETTRANS** will create the dump file and log all activity between **NETTRANS** and the Synchronous Network Engine in use. This information will include all Intrinsic calls in addition to all data transfers. If the dump file already exists, it will be purged and re-built.

To examine the contents of the dump file, you must use the **FMTDUMP** program, accessed via the **#FMTDUMP** command available from within the Network/S application, **NETWORK**. **#FMTDUMP** will attempt to open a dump file in the logon group and account. If the dump file is in another location, or has been renamed, the filename can be specified in the ***dumpinfo*** parameter (below).

For example, to examine a local dump file, enter:

```
#FMTDUMP
```

Parameters to **#FMTDUMP** include:

J	Display formatted listing on **$STDLIST**, 23 lines a time.
L	Direct output to the line printer
S	When specified, **#FMTDUMP** will prompt the user for a search string and a count, *n*, and will start generating output when the *n*'th occurrence of the string is found.
T	Each entry in the formatted output has a time-stamp marking it. If this option is specified, **#FMTDUMP** will prompt for a target time-stamp to search for before beginning the report. The time-stamp value may be specified as an integral number of milliseconds or as seconds.fraction. The presence of a "." in the response indicates the form chosen. If the listing is directed to the line printer, **#FMTDUMP** will prompt for an ending time-stamp as well.

U	Generate a trace listing in User Format, showing the transfer dialog in a format more closely resembling the actual bi-sync transmission.
"*dumpinfo*"	Specify a file other than **XFERDUMP** or a location other than the logon group/account.

The parameters may be combined, separated by commas (,).

Note: Unless displayed in User Format, the trace listing will be difficult to interpret. When necessary, detailed trace analysis can be performed by Telamon, using files created on your system.

To examine a dump file that has been renamed to **OLDDUMP**, enter:

```
#FMTDUMP,"OLDDUMP"
```

To examine a dump file that is located in **PUB.SYS**, additionally specifying the Jump option, enter:

```
#FMTDUMP,J,".PUB.SYS"
```

The output from **#FMTDUMP** will be directed to **$STDLIST** unless the **L** option has been specified. e.g.

```
#FMTDUMP,L
```

will cause the formatted output to be sent to device class **LP** via formal designator **XFERLIST**. If you wish to redirect the output to another device, enter:

```
:FILE XFERLIST;DEV=...
```

before executing **#FMTDUMP**.

The **TRACE** capability is left enabled until **NETTRANS** terminates.

In the absence of an explicit **TRACE** setting, **NETTRANS** will open a temporary, circular trace file, named **XFER0DMP**, which will be saved only in the event of an abnormal program termination. When, and if, this file is saved, an appropriate message will be displayed. Setting **TRACE=NO** or setting the **NORJETRACE** job control word to 1 suppresses the creation of either type of trace file.

This file can then be analyzed using:

```
#FMTDUMP,"XFER0DMP"
```

or

```
#FMTDUMP,0
```

Syntax:

```
#TXLINE ... ;TRACE={YES | NO}
```

Default:

```
#TXLINE ... ;TRACE=NO
```

#TXSTAT

For Engine version 6.5 and higher. This command will generate a transaction reporting the cummulative counts of bi-sync retry/re-transmission indicators. The reported items include:

r_enqs	Received ENQs
r_naks	Received Negative Acknowledgments
r_ttds	Received Temporary Text Delays
r_waks	Received Wait Acknowledgments
x_enqs	Transmitted ENQs
x_naks	Transmitted Negative Acknowledgments
x_ttds	Transmitted Temporary Text Delays
x_waks	Transmitted Wait Acknowledgments
retry	Asynchronous retransmissions

The transaction is sent as Message #9999 in the following format:

```
#9999xr_enq=n r_nak=n r_ttd=n r_wak=n x_enq=n x_nak=n x_ttd=n x_wak=n retry=n
```

where *x* is the Online/Offline indicator (if **STATUS=YES** has been set in the **#TXLINE** command). The counters are reset after each report.

Discussion

Initialization

NETTRANS utilizes the **RJECLINE** initialization file, described earlier, and ignores any options that don't apply.

Reliable Mode

In normal operation, the Network Engine acts as a buffer between the Local System and the external synchronous device. As regards the transmission of data from the Local System, there are two distinct processes at work in the Engine: one to receive records from **NETTRANS** and another to transmit those records, using the protocol specified in the **#TXLINE** command, to the external synchronous device. These processes are only loosely connected to each other. It is possible, for example, for **NETTRANS** to "get ahead" of the Engine by having sent records faster than the Engine's synchronous process can transmit them. This can occur for a variety of reasons, not all of which indicate an error condition. In **3780** contention mode, it is possible that the Engine is in the process of receiving data from the remote system at the same time it is receiving data from **NETTRANS** - in this mode, the Engine must wait until the remote system has ceased transmitting before the Engine can transmit it's data.

In any case, the Engine manages this buffering operation correctly, maintaining the order of data sent. A problem arises, however, if one of the buffered records is rejected by the remote system for any reason. The Engine passes a status result back to **NETTRANS**, informing the program of a Transmit Rejected event, merely indicating that *some* transaction from the transmit queue has been rejected by the remote system - we have no idea, unfortunately, which record was rejected.

Reliable mode provides the capability to monitor each transaction's progress, recording the corresponding acknowledgment or rejection. When this mode is enabled, a two-byte identification field needs to be added to each data record sent to the socket. This ID field is removed from the data before it is sent to the remote system. As each record is acknowledged, the Engine sends a specially formatted packet to **NETTRANS** identifying the record, using the ID field originally included with the record. Additionally, a packet is sent in the event that the remote system rejects the record, similarly identifying the rejected record. In **3270** mode, these packets contain the **CU** and **DA** of the corresponding record along with the ID bytes. In **3780** mode, the **CU** and **DA** fields are blank.

The two event transactions, **ACK** and **NAK**, are written to the socket using message numbers **0101** and **0102**, respectively. The text for these two "messages" consists of the **CU**, **DA** and the 2 original ID bytes.

In **3270** mode, Reliable mode also provides information indicating when each addressed **CU** has initially responded (**MASTER=YES**) or polled (**MASTER=NO**) to a poll/select sequence (message **0103**) and a message indicating that the **CU** is not responding/polling (message **0104**).

The ID field immediately precedes the data portion of the record, following any addressing or framing characters. e.g.

```
          1         2
12345678901234567890...
CDii ... Data ...
UAdd
  12
```

for **3270**, and:

```
          1         2
12345678901234567890...
ssii ... Data ...
ppdd
  12
```

for **3780**.

Example

To use **NETTRANS** to communicate with an Engine attached to logical device 101, with the Computer-to-Engine speed at 19200 baud and the following synchronous characteristics:

```
3270 slave
ASCII data
```

```
EBCDIC control codes
Control Unit Address 0xC1 (EBCDIC A) (Index 1)
```

use the following:

```
!JOB jobname,user.account,group
!RUN NETTRANS.NETWORKS.TELAMON; &
!  INFO="#TXLINE 3270;DEV=101,1920; &
!        LINECODE=EBCDIC;TXCODE=ASCII;POLL=1;MASTER=N"
!EOJ
```

Note that with the default values, the above **RUN** command could have been specified as (the **#TXLINE** keyword may be omitted):

```
!RUN NETTRANS.NETWORKS.TELAMON;INFO="3270;DEV=101,1920;POLL=1;MASTER=N"
```

The initial **#TXLINE** command may also be specified by writing it to the **NTINPUT** message file.

Command Summary

```
#TXLINE
  {3270 | 3780}
  [ ;LINECODE={ASCII | EBCDIC} [, CRC | LRC] ]
  [ ;TXCODE={ASCII | EBCDIC | BINARY} ] input/output character set
  [ ;XPARENT={YES | NO} ] send/receive transparent data
  [ ;CONNECT={{DIAL,"phone"[,modem]} | ANSWER} ]
      (modem is one of {DTR | UDS | SADL | V.25[,ASCII] | 801})
  [ ;DEV=["name of communications port"] [,speed of dev in cps] ]
      (default:="",960)
  [ ;WAIT={[minutes][,seconds]} ] time to wait to go online
      (default:=3,0)
  [ ;MASTER={YES | NO} ] primary or secondary contention
  [ ;POLL=I0 [, I1, ... In] ] poll list (3270)
  [ ;RELIABLE={YES | NO} ]
  [ ;LIMIT={CU not responding limit} (3270) ]
  [ ;SLOWPOLL={Slow Poll Interval} (3270) ]
  [ ;MAXTRAN={Maximum Number of Transactions per Bid} ] (3780)
  [ ;IDLE={Poll Idle Timer} ] (3270)
  [ ;BAUDRATE={Engine-provided synchronous clock speed} ]
  [ ;ID={local terminal id} ] (default:=none)
  [ ;REMID={remote terminal id} ] (default:=none)
  [ ;ADDEOL={YES | NO} ]
  [ ;USERFRAME={YES | NO | UNIX} ]
  [ ;ROUTER={run command} ]
  [ ;STATUS={YES | NO} ]
  [ ;LOAD={procedure}({library}) ]
  [ ;FILES={count} ]
  [ ;CRASH={Crash timer, in minutes} ]
  [ ;SYN={SYN character count} ]
  [ ;NAK={NAK limit} ]
```

```
    [ ;ENQ={ENQ limit} ]
    [ ;TTD={TTD limit} ]
    [ ;WACK={WACK limit} ]
    [ ;DIALWAIT={Dial Wait limit} ]
    [ ;TRACE={YES | NO} ] generate diagnostic trace

#TXVERSION      (Return version information)

#TXSTATUS       (Return status information)

#TXSTAT         (Returns counters)

#TXFLUSH        (Flush all transactions)

#TXFLUSHid      (Flush transaction with reliable mode id id)

#TXCLOSE        (Close Network Engine port - drops connection)

#TXEND          (Terminate)
```

Sample Client Application

A sample client application, **DEMO**, is included with **NETTRANS**. **DEMO** interacts between the specified socket and the user's terminal. **DEMO**'s command line options are:

```
:RUN DEMO;INFO="[-C cu] [-D da] [-L group.account]"
```

cu specifies the Control Unit index to be used in **3270** mode

da specifies the Device Address index to be used in **3270** mode

group.account
identifies an alternate location for the **NTINPUT** and **NTOUTPUT** files.

When run with the **cu** and **da** options, **DEMO** will supply the appropriate Control Unit and Device Address characters to data entered from the keyboard. If **-r** is specified, **DEMO** will supply a suitable Reliable ID field as well.

For example, to use **DEMO** in **3780** mode, you could enter (underlined):

```
:STREAM JNETTRAN
  #Jnn
:RUN DEMO
Demo: Interact between $STDINX/$STDLIST and NTINPUT/NTOUTPUT
      Enter "//" or control-Y to terminate
#0077 NetTrans: 01DEC94-17:54:40.9 - Version: A.01.65 - 01DEC94-17:13
input: #TXLINE 3780;USERFRAME
input: #0017 #TXLINE 3780;USERFRAME
input: #0090 Engine Version: A.7.1 (Dual) (Sync) (AutoDial) (8MHz)
input: #0085 01DEC94-17:59:54.8: Online
```

```
input: this is a test
input: $02 is $20 is $20 the $20 reply $03
input:
```

This will result in the transaction “this is a test” to be sent. **DEMO** provides the **stx** and **etx** framing. The responses starting with 4-digit numbers are messages from **NETTRANS** and the message starting with **$02** (**stx**) is a transaction received from the remote system. The **USERFRAME** option is explained below.

A similar example, using **3270** mode and the Reliable Mode option:

```
:STREAM JNETTRAN
  #Jnn
:RUN DEMO;INFO="-C 0 -D 0 -R"
Demo: Interact between $STDINX/$STDLIST and NTINPUT/NTOUTPUT
      Enter "//" or control-Y to terminate
cu[0] da[0]:
#0077 NetTrans: 01DEC94-18:05:20.5 - Version: A.01.65 - 01DEC94-17:13
cu[0] da[0]: #TXLINE 3270;RELIABLE;MASTER=Y;POLL=0;USERFRAME
cu[0] da[0]: #0017 #TXLINE 3270;RELIABLE;MASTER=Y;POLL=0;USERFRAME
cu[0] da[0]: #0090 Engine Version: A.7.1 (Dual) (Sync) (AutoDial) (8MHz)
cu[0] da[0]: #0085 01DEC94-18:06:04.5: Online
cu[0] da[0]: #0103  “
cu[0] da[0]: IDthis is a test
cu[0] da[0]: #0101   ID
cu[0] da[0]: $20 $20 $02 this $20 is $20 the $20 reply $03
cu[0] da[0]:
```

This will result in the transaction “this is a test” to be sent. **DEMO** provides the **stx** and **etx** framing as well as the Control Unit, Device Address and Reliable Mode ID bytes. The responses starting with 4-digit numbers are messages from **NETTRANS** and the message starting with **$02** (**stx**) is a transaction received from the remote system, with the CU and DA characters present as the first two bytes. Note the **0103** message indicating that the remote system has responded to the polling sequence and the **0101** acknowledgment message.

Program Messages

#0001sPacket terminator error (*n*)

The packet protocol between **NETTRANS** and the Network Engine defines a data format for received data. This error can occur if the received data doesn't match this format. **NETTRANS** attempts to recover from this error condition. This is an internal error and should be referred to Telamon if it recurs.

#0002s*timestamp:*Communications problem: Engine was reset [(hardware)]

An unexpected reset occurred either due to either an internal software problem, the Crash Timer (**RJECLINE CRASH**=) having expired, the reset paddle switch being pressed or the Network Engine's power being disconnected.

The “hardware” suffix indicates that a hardware reset occurred. If no one has reset the Engine via the paddle reset switch or by removing the Engine's power supply, and no power interruptions have occurred, this may indicate that the Engine's power supply is on the verge of failure. Contact Telamon.

#0003s*timestamp:*Communications problem: line dropped
#0003s*timestamp:*Invalid {*type*} {Response | Sequence}

The remote system disconnected unexpectedly (first form.)

When **NETTRANS** issues command sequences to the Network Engine, specific responses are expected. This error indicates that the expected response was not received. Internal software error - contact Telamon.

type can be one of:

SendData[SOH]
SendData[STX]
Status
SetIdleTimer
SetCrashTimer
SetDataLength
SetNAKs
SetENQs
SetTTDs
SetDialWait
SetWACKs
SetDataTimer
SetRTStimer
CancelBid
EndOfProc
Reset
TransmitBid
ReceiveBid
EndOfData
GoToSleep
LoadTable
Flush

#0005s*timestamp:*Communications problem: line dropped (DSR)

The remote system disconnected unexpectedly.

#0006s*timestamp:*Communications problem: Status: $*xxxx*

An error has occurred, consisting of one or more of the following:

- Xmit rej. Too many NAKs received during a send operation.
- Recv rej. Too many NAKs sent during a receive operation.
- Recv Q Ovfl. Rare. The remote system possibly failed to respond to a Wait Acknowledge (WACK) or sent an unpredictably large block of data.

- Xmit Q Ovfl. Rare. **NETTRANS** has sent more data to the Network Engine than can fit in the Engine's transmit queue.
- RVI [after ETX]. A Reverse Interrupt (**RVI**) has been received. The Network Engine will stop its current send operation and await data from the other system.
- Context error. This message indicates that data have been sent to the Network Engine that contain bisync control characters (**STX**, **ETX**, **ETB**, etc.) If the **XPARENT** option hasn't been set, do so. If it *has* been set, this indicates an internal error in the software - contact Telamon.

The $*xxxx* value represent the status bytes indicating this condition.

#0007s[Communications problem:]not online[yet]

Data cannot be sent to the Engine until it reports being online.

#0008sInvalid or Unknown command

An unrecognized command was read from **NTINPUT**.

#0009sCOMMAND Intrinsic error *err* (*parm*)

An error occurred executing the **COMMAND** intrinsic. *err* and *parm* are the two return parameters from the intrinsic.

#0010sCommunications problem: NETTRANS terminating
#0010sCommunications problem: remote went offline

The remote system disconnected unexpectedly.

#0011sPacket length error (*m*/*n*)

Packet format error on data received from the Network Engine. The actual count of characters received from the Engine (*m*) does not match the length field (*n*) that precedes each packet. **NETTRANS** attempts to recover from this error condition. This is an internal error and should be referred to Telamon if it recurs.

#0012sFREAD error on MSG file

An MPE file system error occurred while attempting to read a message file.

#0013sFFILEINFO error on DEV file

An MPE file system error occurred while attempting to determine the characteristics of the Engine’s device port

#0014sCREATEPROCESS Intrinsic error n

An error occurred while executing the CREATEPROCESS intrinsic to process the **ROUTER** command.

#0015s@*nnnn* Parameter Error: *message*

An invalid parameter was found in the **#TXLINE** command. The four-digit field following the **@** sign is the offset into the command string where the error was detected. ***message*** can be one of:

Invalid speed
Invalid DEV (<=8 chars)
Invalid protocol
Invalid option
Invalid modem type
Missing ‘,’
Missing quote
Invalid SELECT
Expected YES or NO
Invalid value
Phone # too long (<=n bytes)
Invalid LINECODE
Expected MIN or MAX
Invalid CONNECT parameter
Invalid ID (<=8 bytes)
Invalid CHECKSUM parameter
Filename too long (<=40 bytes)
Invalid TXCODE
Invalid poll data (offset=n)
Invalid poll list value (0-30)
Too many [(or too few)] poll list entries
Missing ‘;’
Invalid MAXTRAN value (1-15)
Invalid option for 3270 mode
Invalid option for 3780 mode
Invalid dial language

#0016sNo #TXLINE has been issued

Data cannot be sent to the Engine if no **#TXLINE** command has been issued.

#0017s *echo of last command*

NETTRANS echos received **#TX** commands with this message.

#0018sFOPEN error on MSG file

An MPE file system error occurred while attempting to open a message file.

#0019sFCLOSE(*parm*) error on XFERDUMP file

An MPE file system error occurred while attempting to close the trace file.

#0020sFOPEN error on XFERDUMP file

An MPE file system error occurred while attempting to open the trace file.

#0021s*timestamp:*snapshot (XFER0DMP[-temp]) [couldn't be] saved

Message indicating that the automatic trace file was created and saved.

#0022sFCONTROL(*parm1*,*parm2*) error on XFERDUMP file

An MPE file system error occurred while attempting to configure the trace file.

#0023sFRENAME error on XFERDUMP file

An MPE file system error occurred while attempting to rename the trace file.

#0024sFWRITE error on XFERDUMP file

An MPE file system error occurred while attempting to write to the trace file.

#0025sIOWAIT error on DEV file

An MPE file system error occurred while attempting to complete I/O on the Engine's serial port.

#0026sMessage is too short - must be > *n* bytes

At a minimum, a data transaction must contain at least one data character, not including the **stx** and **etx** framing characters or the **CU** and **DA** bytes and one data character in **3270** mode.

#0027s*timestamp:*Poll Idle

This message is generated when the **IDLE** option has been specified and the indicated Poll Idle timer has expired.

#0028sModem dialing error

This message indicates that the auto-dial modem has reported an error during a dial sequence.

#0029sFOPEN error on NTINPUT file

An MPE file system error occurred while attempting to open the **NTINPUT** file.

#0030sFFILEINFO error on NTINPUT file

An MPE file system error occurred while attempting to determine the characteristics of a message file.

#0031sFCLOSE(*parm*) error on NTINPUT file

An MPE file system error occurred while attempting to close the **NTINPUT** file.

#0032sFREAD error on NTINPUT file

An MPE file system error occurred while attempting to read a message file.

#0033sFOPEN error on NTOUTPUT file

An MPE file system error occurred while attempting to open the **NTOUTPUT** file.

#0034sFFILEINFO error on NTOUTPUT file

An MPE file system error occurred while attempting to determine the characteristics of a message file.

#0035sFCLOSE(*parm*) error on NTOUTPUT file

An MPE file system error occurred while attempting to close the **NTOUTPUT** file.

#0036sFWRITE error on NTOUTPUT file

An MPE file system error occurred while attempting to write to the **NTOUTPUT** file.

#0037sFWRITE error on DEV file

An MPE file system error occurred while attempting to write to the Engine's serial port.

#0038sFCONTROL(*parm1*,*parm2*) error on DEV file

An MPE file system error occurred while attempting to configure the Engine's serial port.

#0039s*timestamp:*Communications problem: too many retries

Whenever a format or timeout error occurs between the Network Engine and the **NETTRANS** process, the packet is re-sent up to 4 times. If the error persists, the process terminates. Contact Telamon.

#0040sFREAD error on DEV file

An MPE file system error occurred while attempting to read the Engine's serial port.

#0041sFOPEN error on DEV file

An MPE file system error occurred while attempting to open the Engine's serial port.

#0045sFSETMODE error on DEV file

An MPE file system error occurred while attempting to configure the Engine's serial port.

#0051s{*fserror text*}

The text of the most recent MPE file system error.

#0056sFCONTROL(*parm1*,*parm2*) error on MSG file

An MPE file system error occurred while attempting to configure a message file.

#0058sFFILEINFO error on MSG file

An MPE file system error occurred while attempting to determine the characteristics of a message file.

#0059sInvalid characteristics on MSG file

NETTRANS should be allowd to create any message files it needs. This error indicates that on of the necessary message files was built with an invalid characteristic. Purge the file and allow **NETTRANS** to re-create it.

#0066sNot on poll list

A **3270** record has been received from the message file with a control unit identifier that hasn't been specified in the **#TXLINE...POLL=** sequence.

#0068sToo many consecutive FileSys errors

Whenever an asynchronous transmission error occurs between the Network Engine and the **NETTRANS** process, the packet is re-sent up to 4 times. If the error persists, the process terminates. Contact Telamon.

#0069sInvalid CU

A **3270** transaction has been received from the message file with an invalid Control Unit identifier.

#0070sInvalid DA

A **3270** transaction has been received from the message file with an invalid Device Address identifier.

#0071sFCLOSE(*parm*) error on MSG file

An MPE file system error occurred while attempting to close a message file.

#0074sUnknown file from IOWAIT

The IOWAIT intrinsic returned an unexpected file number. Contact Telamon.

#0077s{*Program version info*}

#0078sNew MSG file width inconsistent with others

All of the message files used by **NETTRANS** must have identical record widths.

#0081sROUTER process terminated

The **ROUTER** process has terminated unexpectedly.

#0084s*ch*/*rec* bytes/recs sent, *ch*/*rec* bytes/recs recvd
#0084s-Queues: Recv (*curr*[*max*]) Xmit (*curr*[*max*])

The result of the **#TXINFO** command.

#0085s*timestamp:*Online

Indication of on-line status - **DSR** went On

#0086sCCG from TXE - close and re-open

An unexpected end-of-file condition was detected on the Engine's serial port. **NETTRANS** will close and re-open the port in an attempt to correct the condition.

#0087sMust be in the form: SOH ad1 ad2 STX data ETX
#0087sMust start with STX/SOH and end with ETX/ETB

Each data record received from the client process must begin with either a Start-of-Text (**stx**) or a Start-of-Header (**soh**) and must be terminated with either an End-of-Text (**etx**) or an End-of-Block (**etb**).

#0088s*timestamp:*Recoverable FSERR *err* (*text*)

Indicates that a recoverable asynchronous I/O error has been detected and that **NETTRANS** is attempting to recover.

#0089s*timestamp:*No response from Network Engine

The Network Engine is not responding to commands from the **NETTRANS** process. **NETTRANS** attempts to recover from this event. If it fails 4 times consecutively, the process terminates. If this error occurs during the execution of the **#TXLINE** command, check the configuration, particularly the speed setting in the **#TXLINE** command. Ensure that this setting matches the switch settings on the Network Engine. If this error occurs later in the command sequence, a hardware failure may have occurred.

#0090s{*Engine firmware info*}

#0091sInvalid ROUTER (<= *n* bytes)

The information specified with the **ROUTER** keyword is too long. *n* indicates the maximum allowable size.

#0092sPoll list must be specified for [BPS | 3624 | 3270] mode

When a polling protocol has been requested, at least one poll unit must be specified.

#0093sOnly 1 POLL entry may be specified if MASTER=NO

This error only occurs with Engines with firmware older than A.5.2, where this restriction holds.

#0094sInvalid FILES parameter: (0 - *n*)

Too many files were specified with the **FILES** option.

#0095sInvalid LOAD (<= 36 bytes)

The value specified with the **LOAD** option must be less than 36 bytes long.

#0096sLoader error #*n*

A loader error occurred while attempt to LOADPROC the procedure specified in the **LOAD** option.

#0097sInvalid library (G|P|S|GX|PX)

For CM processes, the library parameter must be one of G (logon group), P (PUB.logon account), S (PUB.SYS), GX (same group as **NETTRANS**) or PX (PUB group of **NETTRANS**' account.)

#0098s*filename* opened

Reports the opening of each file identified with the **FILES** option.

#0099sIOWAIT error on NTCNTRL file

An IOWAIT error occurred on the control message file

#0100sMust be even number of entries

For custom polling, there must be an even number of poll entries specified.

#0101s*cu da id1 id2*

Reliable mode Acknowledgment response (ACK). ***cu*** is the Control Unit, ***da*** is the Device Address (both blank for **3780** mode) and ***id id*** represents the 16-bit id field included with the original data. (Blanks between fields are included here for clarity.)

#0102s*cu da id1 id2*
#0102sNAK: '*cuda*'

Reliable mode Negative Acknowledgment response (NAK). ***cu*** is the Control Unit, ***da*** is the Device Address (both blank for **3780** mode) and ***id id*** represents the 16-bit id field included with the original data. (Blanks between fields are included here for clarity.)

#0103s*cu da*
#0103sRSP: '*cuda*'

Reliable mode Unit Responding message (**3270** only). ***cu*** is the Control Unit, ***da*** is the Device address, although this message only reports activity on a Control Unit basis. (Blanks between fields are included here for clarity.)

#0104s*cu da*
#0104sNOR: '*cuda*'

Reliable mode Unit Not Responding message (**3270** only). ***cu*** is the Control Unit, ***da*** is the Device Address, although this message only reports activity on a Control Unit basis. (Blanks between fields are included here for clarity.)

#0105s*cu da id1 id2*

Reliable mode ID Flushed response. Sent when a **#TXFLUSH*id*** command was sent. For **3270**, ***cu*** is the Control Unit, ***da*** is the Device Address; otherwise both positions are blank. ***id1 id2*** identifies the ID of the deleted transaction. (Blanks between fields are included here for clarity.)

#0106sAll Flushed

Reliable mode All Flushed response. Sent when the Engine has performed the **#TXFLUSH** command.

#0107sFlush failed: *cu da id1 id2*

Reliable mode Flush failure response. Sent when a **#TXFLUSH*id*** command was sent. For **3270**, ***cu*** is the Control Unit, ***da*** is the Device Address; otherwise both positions are blank. ***id1 id2*** identifies the ID of the first transaction that was in the queue, contrasted with the ID that was requested. (Blanks between fields are included here for clarity.)

#0108sWas Empty

Reliable mode Flush empty response. Sent when the Engine tried to perform the **#TXFLUSH** or **#TXFLUSH*id*** command, but the transmit queue was empty.

Modem Call Progress Messages

UDS - Universal Data Systems

```
(E) [UDS] Error in dial message {fatal}
(N) [UDS] No dial tone detected {fatal}
(D) [UDS] Dial tone detected
(R) [UDS] Ring back tone detected
(B) [UDS] Busy line detected {fatal}
(T) [UDS] Re-order tone detected {fatal}
(Q) [UDS] Quit (Abort) timeout {fatal}
(A) [UDS] Call complete
(I) [UDS] Invalid answer tone detected {fatal}
(?) [UDS] Unknown response
```

SADL - Synchronous Auto Dial Language

```
(D) [SADL] Dialing
(R) [SADL] Ringing
(A) [SADL] Answer tone
(L) [SADL] Online
(B) [SADL] Busy {fatal}
(F) [SADL] Failed call {fatal}
(E) [SADL] No dial tone detected {fatal}
(C) [SADL] Invalid dial command {fatal}
(?) [SADL] Unknown response
```

V.25

```
(VAL) [V.25] Valid command
(CNX) [V.25] Connected
(INV) [V.25] Invalid command {fatal}
(CFIET) [V.25] Engaged tone (busy signal) {fatal}
(CFICB) [V.25] Modem busy {fatal}
(CFIRT) [V.25] Ring tone collision {fatal}
(CFIAB) [V.25] Call aborted {fatal}
(CFINT) [V.25] No dial tone {fatal}
(???) [V.25] Unknown response
```

801 Autodialers: UDS801A/C (801)

```
(P) [801] Power Failure {fatal}
(N) [801] Dialing (PND)
(D) [801] Modem in use (DLO) {fatal}
(B) [801] Busy/No answer (ACR) {fatal}
(A) [801] Quit (No COS/DSC) {fatal}
(C) [801] Answer (COS)
(E) [801] Dialing Error (ACR) {fatal}
(???) [801] Unknown response
```

Files

NTINPUT	Input message file.
NTOUTPUT	Output message file.
NTOUTP*nn*	Alternate output message file(s) when **FILES** is specified.

Glossary

RS-232 Modem Signals (Subset)

Mnemonic	Description	Pin	CCITT	EIA
FG	Frame Ground	1	101	AA
TD	Transmit Data	2	103	BA
RD	Receive Data	3	104	BB
RTS	Request To Send	4	105	CA
CTS	Clear To Send	5	106	CB
DSR	Data Set Ready	6	107	CC
SG	Signal Ground	7	102	AB
DCD	Data Carrier Detect	8	109	CF
TC	Transmit Clock	15	114	DB
RC	Receive Clock	17	115	DD
DTR	Data Terminal Ready	20	108.2	CD
RI	Ring Indicator	22	125	CE
(TC)	Ext. Transmit Clock	24	113	DA

BSC Control Characters

Mnemonic	Description	EBCDIC	ASCII
syn	Synchronous Idle	'32'	'16'
soh	Start-of-Header	'01'	'01'
stx	Start-of-Text	'02'	'02'
etx	End-of-Text	'03'	'03'
eot	End-of-Transmission	'37'	'04'
enq	Enquire	'2D'	'05'
dle	Data-Link-Escape	'10'	'10'
dc1	Device-Control-1	'11'	'11'
dc2	Device-Control-2	'12'	'12'
dc3	Device-Control-3	'13'	'13'
esc	Escape	'27'	'1B'
nak	Negative-Acknowledge	'3D'	'15'
etb	End-of-Text-Block	'26'	'17'
gs	Group-Separator	'1D'	'1D'
rs	Record-Separator	'1E'	'1E'
us	Unit-Separator	'1F'	'1F'
ttd	Temporary Text Delay	stx enq	stx enq
ack0	Acknowledge 0	dle '70'	dle 0
ack1	Acknowledge 1	dle /	dle 1
wack	Wait Before Transmit	dle ,	dle ;
rvi	Reverse-Interrupt	dle @	dle >
dle-eot	Switched Line Disconnect	dle eot	dle eot