

BLAST[®]

Professional DOS User Manual

BLAST[®]
SOFTWARE

The File Transfer Authority

©2000 by BLAST SOFTWARE, INC.
49 Salisbury Street West
Pittsboro, NC 27312
All Rights Reserved

Manual #2MNPDOS
6/00

The information in this manual has been compiled with care, but BLAST, Inc., makes no warranties as to accurateness or completeness, as the software described herein may be changed or enhanced from time to time. This information does not constitute commitments or representations by BLAST, Inc., and is subject to change without notice.

BLAST® is a registered trademark, and BLAST Professional™, BLAST Professional UNIX™ and TrueTerm™ are trademarks of BLAST, Inc. Any trademarks, trade-names, service marks, service names owned or registered by any other company and used in this manual are proprietary to that company.

Restricted Rights Legend

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subdivision (b) (3) (ii) of the Rights in Technical Data and Computer Software clause at 52.227-7013.

BLAST, Inc.
49 Salisbury Street West
P.O. Box 818
Pittsboro, North Carolina 27312

SALES: (800) 242 - 5278
FAX: (919) 542 - 0161
E-mail: info@blast.com
Technical Support: (919) 542 - 3007
E-mail: support@blast.com
World Wide Web: <http://www.blast.com>

© Copyright 2000 by BLAST, Inc.

Table of Contents

1 Introduction 1

BLAST Software Registration	1
The BLAST Package	2
BLAST Professional Features.	2
How To Use This Manual.	3
BLAST Technical Support	5

2 The BLAST Environment 7

Introduction.	7
Environment Variables	8
Command Line Switches	10
BLAST in the Background	13
BLAST.OPT	19
BLAST.OPT Settings	20
Using BLAST on a LAN	28
Flow Control.	35

3 BLAST Quickstart 37

Starting BLAST	37
The BLAST Screen.	38

Three Keys to Remember	40
The BLAST Menu.	41
A Quickstart File Transfer	42

4 The Menu 49

Moving Through the Menu	49
The Keyboard	50
Using a Mouse	51
The Offline Menu	52
The Online Menu	54
The Filetransfer Menu.	56
The Local Menu	57
The Remote Menu.	58
Automation with BLASTscript.	59

5 The Setup 61

What is a Setup?	61
Setup Fields.	65
ANSI Emulation Subwindow	72
DG Emulation Subwindow.	73
DEC VT Emulation Subwindows.	74
WYSE Emulation Subwindows	78
HP Emulation Subwindow	82
BLAST Protocol Subwindow	87
Kermit Protocol Subwindow.	92

Zmodem Protocol Subwindow	95
-------------------------------------	----

6 BLAST Session Protocol 101

The BLAST Session Protocol.	102
BLAST Protocol Design.	103
Starting a BLAST Session	105
Ending a BLAST Session	108
Performing Filetransfer Commands	110
Transfer Command File	117
BLAST Protocol Remote Menu	120
Automating the BLAST Session Protocol	121
Fine-Tuning the BLAST Session Protocol	121
Filetransfer Security with BLAST Protocol	123

7 FTP 125

Using FTP.	125
Starting an FTP Session	126
FTP Filetransfer Menu	126
Sending and Receiving Files with FTP.	127
Filenames Restrictions with FTP	128
Ending an FTP Session.	129
FTP Commands	129

8 Kermit Protocol 131

Kermit Filetransfer Menu	131
Sending and Receiving Files with Kermit	132
File Transfer Switches with Kermit	134
Filenames Restrictions with Kermit	135
Kermit Remote Menu	135

9 Xmodem, Ymodem, and Zmodem Protocols 139

Command Line Features	140
Xmodem Protocol	140
Ymodem Protocol	141
Zmodem Protocol	142
Filenames Restrictions	143

10 Text Transfers 145

Introduction	145
Uploading Text to a Remote Computer	145
Downloading Text from a Remote Computer	146

11 BLAST Editor 149

Using BLAST Editor	149
Quick Reference	150
Cursor Movement and Scrolling	151

Inserting and Deleting Text	151
Managing Text Blocks	152
Searching.....	152
Quitting BLAST Editor.....	153

12 Introduction To Scripting **155**

Starting Out.....	155
Learn Mode.....	160

13 BLASTscript Topics **165**

Scripting Basics	165
Manipulating Text.....	170
Managing the Screen Display.....	174
Communicating with Other Programs	176
File Transfers with BLAST Session Protocol.....	178
File Transfers with FTP	181
File Transfers with Kermit	181
File Transfers with Xmodem and Xmodem1K.....	184
File Transfers with Ymodem and Ymodem G	185
File Transfers with Zmodem.....	187
Using Log Files for Error Checking.....	188
Text Transfers.....	190

14 Connecting and Disconnecting **193**

Introduction. 193
BLASTscript Libraries 193
The Index Utility. 199

15 BLASTscript Command Reference **201**

Introduction. 201
Data Types 201
Syntax Rules 204
Commands That Set @STATUS 204
BLASTscript Statements 205

16 BLASTscript Reserved Variables **239**

17 Data Stream Control and Terminal Emulation **283**

Introduction. 283
Data Stream Filtering and Alteration 283
Terminal Emulation 287
Keyboard Mapping Utility 296

18 Remote Control **303**

What Is Remote Control? 303

Connecting to the Host PC	304
Taking Control	306
Online Menu Options	306
Transferring Files to and from the Host PC	307
Disconnecting from the Host PC	308
Using Access Mode	308
Using Terminal Mode.....	311
Modifying BHOST Settings	312

Appendix A Error Messages **321**

Introduction.....	321
BLAST Protocol Functions.....	321
Transfer File Management	322
Utility File Management.....	323
Scripting	323
Initialization	324
Script Processor	325
Network.....	325

Appendix B Key Definition Charts **327**

BLAST Keys.....	327
Terminal Emulation Keys.....	329

Appendix C Troubleshooting 337

Installing BLAST 337
Starting BLAST 338
Going Online. 338
File Transfer 340

Appendix D The ASCII Character Set 341

Appendix E Autopoll 343

The Autopoll Script. 343
Installing Autopoll 344
Starting Autopoll. 344
The Site File 346
Transfer Command File 347
Overview of Autopoll Script Actions 347
Configuration Example. 349
Other Files Using the Filename Stub 351
Tips and Tricks 353
Modifying Autopoll 355
Configuration Worksheets 357

Appendix F Glossary 361

Chapter 1

Introduction

BLAST Software Registration

Thank you for buying our communications software and welcome to the world of BLAST. Before doing anything else, it is *very* important that you complete the Warranty Registration Card. Without it, we cannot provide you with the complete support and continued service that comes with every copy of BLAST.

The services available to registered owners of BLAST include:

- ◇ A ninety-day warranty stating that the software will operate according to specifications in effect at the time of purchase.
- ◇ Professional help from our experienced Technical Support staff for a nominal fee.
- ◇ New product announcements.
- ◇ Discounts on product upgrades.

Extended warranties, custom support, special training, and corporate licensing are also available. Please call BLAST, Inc. at (919) 542-3007 or refer to the enclosed literature for more information.

The BLAST Package

The BLAST package contains the following items:

- ◇ Two 3-1/2" diskettes containing the BLAST and BHOST programs.
- ◇ One BLAST Professional License Agreement and Warranty found on the front of the diskette package. It is important to read and understand the terms and conditions in this document before opening the package.
- ◇ One Warranty Registration Card. The serial number of your BLAST program is printed on this card. When placing a call to BLAST Technical Support, please have this number available. Also, please read the card, fill it out, and send it immediately to BLAST, Inc.
- ◇ The following BLAST documentation: an Installation Guide, User Manual, Quick Reference card, Quick Start card, and *BHOST User Manual*.

If the package does not contain all of these items, please call the BLAST Customer Support staff.

BLAST Professional Features

BLAST Professional is designed to connect your IBM PC or compatible MS-DOS computer to a variety of other computers. You may use one of the following connections:

- ◇ any asynchronous modem and regular phone lines
- ◇ TCP/IP
- ◇ hard-wired cables
- ◇ X.25 or ISDN networks and other virtual asynchronous circuits

BLAST Professional transfers files to and from remote computers with the fast and 100% error-free BLAST protocol. You may also choose from BLAST's implementation of FTP, Kermit, Xmodem, Ymodem, and Zmodem protocols.

BLAST Professional features remote control that allows one PC to take complete control of another PC. Remote control works over modems or through a LAN and includes full remote mouse support, automatic translation between different video modes, password-protected dial-back security, and a host of other features.

BLAST Professional allows your PC to emulate many popular terminal types, including the VT320 and Wyse 60, for interactive work on DEC, UNIX, Xenix, Data General, and Hewlett-Packard systems, as well as other multi-user computers.

BLAST Professional also includes other advanced features such as data compression, scripting, and keyboard remapping.

How To Use This Manual

Parts of the Documentation System

Each portion of the BLAST documentation system fulfills a specific need:

- ◇ Online Help is always available while you are using BLAST. It is context-sensitive so that the information you need is right at hand. To access Online Help, press F1.
- ◇ The Installation Guide contains step-by-step instructions for installing and configuring BLAST.
- ◇ The User Manual contains all the information necessary for operating BLAST, including detailed descriptions of Terminal mode and filetransfer procedures. It also contains general information as well as a listing of all BLAST functions, BLASTscript reserved variables, and BLASTscript statements. The listing for each BLASTscript statement includes syntax, usage details, and examples.
- ◇ The Quick Reference Card is a handy list of BLASTscript commands, BLAST Keys, command line parameters, and more.

- ◇ The Quick Start Card is a condensed BLAST Professional tutorial for experienced users.
- ◇ The *BHOST User Manual* guides you through installing and configuring BHOST, the remote control host program for DOS-based PCs.

Manual Overview

If you are experienced with telecommunications, you may not need to read the entire manual to learn how to perform specific tasks with BLAST. Following is a quick guide to finding important information in this manual:

<u>If you need information about</u>	<u>Look in</u>
General BLAST operation	Chapter 3
Creating keyboard maps	Chapter 17
How to use the BLAST protocol	Chapter 6
How to use FTP	Chapter 7
How to use Kermit	Chapter 8
How to use Xmodem, Ymodem, & Zmodem	Chapter 9
Taking remote control of another PC	Chapter 18
Command line options	Chapter 2
Communications port assignments	Chapter 2
Installing network drivers	Chapter 2
Scripting	Chapter 12
Modifying BLAST's support files	Chapter 14
Troubleshooting	Appendix C

Documentation System Conventions

To help reduce confusion, all BLAST documentation shares several common name conventions, display conventions, and defined terms:

- ◇ Examples in the text indicate the actual keystrokes you should type to perform a function. For example:

```
send myfile.txt ENTER
```

instructs you to type “send myfile.txt” and then press the ENTER key. In early introductory chapters, “ENTER” is included to indicate the keystroke needed to execute input of typed data. In later chapters, it is assumed and omitted.

- ◇ Italics in code indicate that the item (for example, a command line argument or a string value) is generic and that a more specific item is needed. For example, in the following lines of code,

```
Connect
Filetransfer
Send
local_filename
remote_filename
to
esc
```

specific filenames should be given for *local_filename* and *remote_filename*. An exception to this convention is the all-italic format used for command descriptions in Chapter 15.

- ◇ The term “local” computer refers to the machine closest to you, whereas “remote” computer refers to the system to which your local machine is connected.
- ◇ The term “interactive” describes BLAST operation from the keyboard. When operating interactively, a user presses keys to control the program. Alternatively, a user may write a BLAST script to control the program.
- ◇ Finally, “Terminal mode” describes BLAST operation as a terminal to a remote computer. For example, if you are going to use BLAST to connect to a remote Data General computer and you have selected the D200 keyboard emulation in BLAST, then your keystrokes will be interpreted by the DG computer as if you were operating from a DG terminal.

Comments and Suggestions

Considerable time and effort have been spent in the development of this product and its documentation. If you are pleased, or not pleased, we would like to hear from you. Please send us your comments and suggestions. For your convenience, a FAX reply form is provided at the back of this manual.

BLAST Technical Support

If you have problems installing or running BLAST, first look for answers in your manuals and in the Online Help. Double-check your

communications settings, operating system paths, modem cables, and modem power switches.

If you are still unable to resolve the problem, contact BLAST Technical Support. For a nominal fee, a technician will help you with your problem. Technical Support may be purchased on a per-incident basis or annually. Contact our Sales Staff for details. If you purchased BLAST outside of the USA, please contact your authorized distributor for technical support.

What You Will Need To Know

Before you contact us, please have the following information ready:

- ◇ Your BLAST version number and serial number. These numbers appear in the opening banner (when you first start BLAST), in the Online Help window, and on your distribution diskettes.
- ◇ Your operating system version number (e.g., MS-DOS 6.1). To display your version number, type “VER” at the command line.

How to Contact Us

Telephone support is available Monday through Friday. If voice support is inconvenient, you may FAX or e-mail your questions to BLAST, 24-hours-a-day. Please see the title page of this manual for contact numbers and addresses, and the pages at the end of the manual for a sample FAX cover sheet.

Chapter 2

The BLAST Environment

Introduction

BLAST operates in a complex environment. It is not unusual for a computer to be connected to a modem, mouse, printer, network interface, and other hardware. BLAST must work smoothly with this equipment and with other software programs loaded on your system. Several features of BLAST are designed to help you integrate BLAST into your computing environment. These features include support for:

- ◇ DOS environment variables
- ◇ Command line switches
- ◇ Background operation
- ◇ Runtime configuration file (BLAST.OPT)
- ◇ Network operation

Environment Variables

When BLAST and BHOST are executed, they check the DOS environment for the existence of special variables. If one of these variables has been defined, its value will override settings that have been written into the executables or specified in the BLAST.OPT file (see “BLAST.OPT” on page 19).

To set a DOS environment variable, type:

```
SET VARIABLE=VALUE
```

at the DOS command line. Note that there are no spaces on either side of the equal sign.

To set the variable BLASTDIR to the value C:\BLAST, for example, use the command:

```
SET BLASTDIR=C:\BLAST
```

You may also include in your AUTOEXEC.BAT a command of this format. For information on modifying AUTOEXEC.BAT, refer to your MS DOS manual. Following is a list of environment variables that BLAST and BHOST check for:

BLASTDIR

Specifies the directory where BLAST looks for BLAST.HLP, MODEMS.SCR, and SYSTEMS.SCR, and where BHOST looks for MODEMS.SCR, its accounts file, and its log file.

Note that the BHOST accounts file and log file may also be independently specified via BLAST.OPT with ACTFILE and LOGFILE assignments.

The value of BLASTDIR is written to the BLAST and BHOST executables during installation. It can be changed as shown above and through BLAST.OPT with a BLASTDIR assignment.

OPTDIR

Specifies the directory where BLAST and BHOST look for the BLAST.OPT file, which is normally located in BLASTDIR. By setting OPTDIR to a directory containing an alternative BLAST.OPT file, you may temporarily override existing BLAST.OPT settings.

For example, network users may want to locate a separate BLAST.OPT file in a directory other than BLASTDIR (for more on the use of OPTDIR in configuring LANs, see page 17).

SETUPDIR

Specifies the directory where BLAST and BHOST look for setup files. The value of SETUPDIR is written to the BLAST and BHOST executables during installation. It may be changed with the DOS SET command as explained above or through BLAST.OPT with a SETUPDIR assignment.

TMP

Specifies the directory where BLAST and BHOST store temporary files. In the following circumstances, BLAST stores temporary files:

- ◇ When BLAST is in background mode and is toggled to the foreground via the ALT+H Hot Key, BLAST saves the graphics screens of the application toggled to the background.
- ◇ When Access mode is temporarily suspended, BLAST saves the graphics screens of the Host PC.
- ◇ When the BLAST.OPT VIDEOBUF is set to DISK, text screens are also saved.

BHOST saves graphics screens when the Control PC user temporarily suspends Access Mode.

If TMP is not defined, temporary files are saved to the directory from which BLAST or BHOST was executed.

NOTE: Setting TMP to a RAM drive greatly improves performance. Also note that the size of the file to be saved varies widely with the active video mode. Text screens require about 4K of space, CGA screens about 16K, and VGA screens up to 256K.

Network users, especially those with 8-bit network cards, should set TMP to a local (non-server) directory or RAM drive.

Command Line Switches

By using command line switches, you can automatically execute certain BLAST functions at startup. For example you can automatically load a setup and run a BLAST script that brings you directly into a communications session without interactive input. You may also suppress screen displays or run BLAST as a memory-resident program in the background. BLAST recognizes the following switches and parameters:

```
blast [setupname] [/sscriptname] [argument] [/b]  
[/c] [/i] [/n] [/px] [/q] [/tx] [/v] [/w] [/x] [/y]
```

One space must precede each switch included on the command line. Do not insert a space between the switch and the parameter associated with it.

setupname

specifies a setup file for BLAST to load. It is not necessary to type the filename extension. If a script is specified as a parameter in the setup, it will be executed automatically. If no script is specified, BLAST will load the setup and display the Offline menu. If a setup is not specified on the command line, BLAST will automatically load the default setup. BLAST looks for setups in SETUPDIR. SETUPDIR may be specified as an environment variable (see preceding section).

/sscriptname

specifies the BLAST script that will control the current session. Control will be passed automatically to the script instead of the regular BLAST menus and will return to the menu system at completion. If a valid BLAST script is named in the Script File setup field, the script specified by the */sscriptname* switch will override the one specified in the setup. No spaces are allowed between */s* and the script name. BLAST looks for scripts in the current directory first and then in BLASTDIR. BLASTDIR may be specified as a DOS environment variable (for details, see preceding section).

argument

specifies one of ten optional arguments (text strings) that can be passed to a BLAST script directly from the command line. These arguments are stored as the BLASTscript reserved variables @ARG0 to

@ARG9. This switch requires that a setup file be specified on the command line. If no setup is specified, BLAST will interpret the first argument as a setup name and will generate an error message if no setup with that name exists.

/b

starts BLAST in background mode, which allows BLAST to reside in resident memory while another program runs in the foreground; using the ALT H Hot Key, the user can toggle between BLAST and the other program (for details, see “BLAST in the Background” on page 13).

/c

specifies that DTR not be dropped when BLAST is exited.

/i

specifies that BLAST bypass checking of ports that may be reserved by other applications.

/n

forces BLAST to execute in no display mode, in which all displays are suppressed. This switch allows you to integrate BLAST into your applications and batch jobs without losing the information previously written to the screen. BLAST scripts may still select portions of the screen to turn on and off (see “Managing the Screen Display” on page 174). BLAST automatically uses TTY terminal emulation in this mode.

/px

specifies the pad character (x) to be used with Xmodem transmissions. The default pad character is a NUL (ASCII 0).

/q

forces BLAST into quiet mode. Audible signals that normally call attention to prompts and errors are suppressed.

/tx

specifies the end-of-transmission (EOT) timeout for Xmodem transmissions in hundredths of seconds—timeout equals $x/100$ seconds. The minimum timeout is .1 second (10) and the maximum is 600 seconds (60000). For example, starting BLAST with the `/t1000` switch specifies a 10 second end-of-transmission timeout.

/v

starts BLAST in expert (non-verbose) mode, in which prompts asking for confirmation to perform an action are suppressed.

/w

with VT320/220 emulation, forces BLAST to use BIOS screen writing routines instead of direct screen writes. Use `/w` with multitasking environments or on PCs that are not 100% IBM PC compatible. Use `/w` only if you have problems with VT320/220 emulation—the BIOS screen writing routines are significantly slower than direct screen writes.

/x

enables Extended Logging, which writes detailed information about file transfers to your session log. Extended Logging may also be enabled with the reserved variable `@XLOG`.

/y

starts BLAST in four-digit year mode. In four-digit year mode, all dates stored in the BLASTscript reserved variable `@DATE` and in log files will have four-digit year codes.

NOTE: For compatibility with current BLAST products, BLAST operates by default in two-digit year mode. If you require four-digit year codes, you must use the `/y` switch. Be aware that turning on four-digit year mode will alter the format of both log files and the `@DATE` reserved variable. If your current scripts or other applications parse log files or the output of the `@DATE` reserved variable based on fixed offsets, they will require modification to work with BLAST in four-digit year mode.

Example Command Line

The example command line shown below starts BLAST in background mode with a setup named DIAL.SU, a script named NEWYORK.SCR, and “\$30” as an argument to be used by the script:

```
blast dial /snewyork /b $30
```

Precedence for Specifying Options

Because the command line can specify options that can also be named in setups and scripts, BLAST follows a well-defined order of precedence:

- ◇ Whenever a command line switch conflicts with a value specified in a setup also loaded from the command line, the command line switch overrides the setup value.
- ◇ Whenever a command line switch conflicts with a setup value that has been loaded after starting BLAST (through interactive command or BLASTscript control), the setup value overrides the command line switch.
- ◇ Whenever a BLAST script changes a value that was specified in either the setup or the command line, the script change overrides the setup or command line value.

BLAST in the Background

Your PC supports both foreground and background applications. When you start BLAST with the `/b` switch, BLAST is forced to run in background mode, allowing the user to perform other tasks and DOS commands in the foreground.

When BLAST is started in background mode, you can switch between BLAST and a foreground program by pressing the `ALT H` Hot Key. Although foreground applications are suspended when you switch to BLAST, BLAST is never suspended. Thus, entire BLAST sessions may run unattended in the background via script automation.

Starting BLAST in Background Mode

To start BLAST in background mode, type

```
blast /b
```

at the command line. If no `GROUND` option in the `BLAST.OPT` file is set or if the `GROUND` option has been set to `BACKGROUND`, BLAST will begin in interactive mode. If `GROUND` is set to `FOREGROUND`, BLAST will start with no display and immediately pass control to the DOS system prompt. To access BLAST in interactive mode, simply press `ALT H`. (For more on `BLAST.OPT`, see “`BLAST.OPT`” on page 19 and “`BLAST.OPT` Settings” on page 20).

Limitations of Background Mode

Some programs may be incompatible with BLAST in the background because they either use the same commport as BLAST or they occupy too much memory. BLAST requires approximately 400K of RAM while running in the background, depending on your `BLAST.OPT` settings (for details, see “Settings for Minimum Memory Use” on page 17).

When you switch from a foreground application to BLAST, BLAST automatically saves the foreground screen so that the screen is restored when you switch back. The amount of memory needed to save the foreground screen varies with the type of display: about 4K for monochrome, 16K for CGA, and up to 256K for EGA/VGA.

If there is not enough memory to store your foreground screen, BLAST automatically attempts to save it in a temporary disk file. Setting the `VIDEOBUF` option in `BLAST.OPT` to `DISK` forces BLAST to store the foreground screen on disk. If there is not enough disk space to store the foreground screen, BLAST beeps three times, signaling you to exit from the foreground application before switching to BLAST.

IMPORTANT: The `ALT H` Hot Key is disabled while BLAST is set to D461 terminal emulation, is set to 132 column compressed mode, or displays graphics in Access mode.

When BLAST is started with the `/b` switch, interactive use of the System and Edit command from the Local menu is not available. Using the `ALT H` Hot Key to toggle to the DOS system prompt, however, you may access system commands and may execute the BLAST editor by typing

BLASTEDT

at the DOS system prompt. Exit the editor as you would interactively.

Removing BLAST from Memory

Because BLAST occupies memory while in background mode, you should remove it after you have finished a communications session. There are three ways to reclaim the extra memory:

1. If you are running BLAST without the video-suppress switch (/n), you may use the ALT H Hot Key to access BLAST in interactive mode and then exit normally.
2. If you are running BLAST with the video-suppress switch, you may use the program BLASTAT.EXE, included with your distribution diskette. At the DOS prompt, type

```
BLASTAT
```

You will be presented with the following options:

```
F1   Terminate BLAST after current session completes
F2   Terminate BLAST immediately
F10  Exit this program
```

Press F1 or F2.

NOTE: With certain installations of Windows 95, BLASTAT.EXE will *not* release BLAST from memory.

3. You may run a BLAST script that ends with a QUIT statement.

IMPORTANT: Any program installed as memory-resident after BLAST must be removed *before* BLAST can be successfully removed from memory. Exiting BLAST before exiting the other memory-resident program will make it impossible for your PC to reclaim BLAST's memory. In effect, your PC cannot "reach" BLAST to remove it.

Adjusting BLAST.OPT Settings for Background Mode

The BLAST.OPT file provides you with an easy way to customize BLAST for efficiency and personal preference. Each time BLAST is run, if a BLAST.OPT file exists, it is read and the values it specifies remain in effect for the entire session. To optimize a particular use of BLAST in background mode, adjust the following BLAST.OPT options: BG_BLK_SIZ, FILEBUF, GROUND,

MEMPOOL, SLICE, and VIDEOBUF. For a detailed description of each option and its settings, see “BLAST.OPT Settings” on page 20.

You will have to experiment with settings to achieve the best results. The following sections offer suggested settings for BLAST.OPT options tailored to specific applications of BLAST in background mode and a method for testing your settings.

Settings for Efficient Use of BLAST in Background Mode

Some users will start BLAST in the background for minimum use—for example, to monitor a communications port or to perform occasional BLAST transfers from a command file. For these users, the efficiency of BLAST is not as important as the efficiency of their foreground operation. Other users run BLAST as their primary task but wish to perform simple operating system commands or run an occasional small application in the foreground. For these users, BLAST in background mode should be run at maximum efficiency. For both these users the following BLAST.OPT settings will improve performance:

- ◇ Increase FILEBUF to 30 or 40. BLAST then uses a larger buffer size for disk operations, cutting down the number of times it must access the PC’s drive.
- ◇ Increase the BG_BLK_SIZ option closer to its limit of 2048. During BLAST transfer, this setting will minimize the amount of time needed to block a packet of data before sending it.
- ◇ Increase the value of MEMPOOL since the total memory overhead required for BLAST will rise after adjusting FILEBUF and BG_BLK_SIZ. Some experimenting will be required to find the best value for MEMPOOL.

Settings for SLICE are different for these two groups of users:

For users making minimal use of BLAST in background mode, SLICE should be set to 10 (or as close to 1 as possible). This setting forces BLAST to use the least amount of processor time, effectively cutting back the time your PC spends working with BLAST.

For users running BLAST as their primary program, SLICE should be set to 1000 (or as close to 9999 as possible). This setting allows BLAST to use more processor time; how much depends on how slowly a user can afford to let the foreground task operate. Note that a faster PC has more power to spare, but foreground applications using graphics typically require a lot of the processor’s attention.

Settings for Minimum Memory Use

Finally, some users have less memory in their PCs to devote to BLAST. Under these conditions, it is best to run BLAST through a script or command file, to use the no-display option, and to make the following BLAST.OPT adjustments:

- ◇ Set VIDEOBUF to DISK. When you use the ALT H Hot Key, BLAST will save the current screen information to disk rather than to memory, freeing 4 to 16 kilobytes of memory.
- ◇ Decrease FILEBUF to 2048, the minimum acceptable. BLAST then uses a smaller memory buffer for disk operations.
- ◇ Decrease BG_BLK_SIZ to 1, the minimum acceptable. During BLAST transfer, less memory will be used to compress and block each packet of data before it is sent. (Note that you cannot set the Packet Size used by BLAST to a value larger than BG_BLK_SIZ).
- ◇ Decrease the value of MEMPOOL since the total memory overhead required for BLAST will drop. Some experimenting will be required to find the proper value for MEMPOOL.
- ◇ Select TTY as your emulator, minimizing the number of buffers required in Terminal Mode.

Considerations for BLAST Configured for LANs

When BLAST is configured during installation to support LANs, the drivers needed for different protocol drivers are written to the BLAST.OPT. Normally, this action does not effect memory usage for BLAST since BLAST loads the driver specified in your current setup only when you go Online; your PC needs only enough memory for the driver you actually use in a session.

In Background mode, however, BLAST must pre-allocate enough space for all the drivers you might use by automatically loading every driver defined in your BLAST.OPT file, making the memory required by these drivers unavailable to your other applications.

A way to prevent this large memory allocation is to load drivers selectively. To load drivers selectively, create multiple BLAST.OPT files in separate directories and then place the appropriate directory in your search path by setting the OPTDIR environment variable to your directory's name. Remember that BLAST adds driver descriptions rather than overwriting them; make sure that other

BLAST.OPT files in your search path do not have other drivers defined.

NOTE: Decreasing the amount of memory BLAST has available may adversely affect your filetransfer throughput. You may regain some speed by disabling compression, which is a CPU-intensive process.

Testing Your Background Settings

Changes to BLAST.OPT may affect:

- ◇ The total memory required for background (the MEMPOOL).
- ◇ The speed of BLAST operation.
- ◇ The efficiency of BLAST transfer.

Therefore, it is important to test your new BLAST.OPT settings to insure both BLAST and your foreground applications operate as they should. After creating or modifying any settings:

- ◇ Perform BLAST tasks in both foreground and background as well as tasks of the foreground application. For BLAST, this may include BLAST scripts, command files, or interactive work using the ALT H Hot Key.
- ◇ Check programs for errors or unacceptable operation. Are your tasks running too slowly? If you are performing a background BLAST transfer session, check the time necessary to complete the transfer.

Of course, memory limitations may cause errors. If BLAST displays the error message

Error allocating memory from the system

your MEMPOOL value is probably too large. However, if the error message

Error allocating memory from the BLAST memory pool

is displayed, your MEMPOOL value is probably too small.

- ◇ When running BLAST scripts in background with limited memory, you may wish to display (or write to disk) the value of

@MAXMEM. This variable holds the highest amount of memory BLAST has used up to the current point in the program.

BLAST.OPT

BLAST.OPT is the BLAST options file. It is used to customize BLAST and BHOST configurations and to specify alternate configurations for multiple users. BLAST.OPT affects background mode, remote control, and global BLAST operations. If a LAN was specified during BLAST installation, a BLAST.OPT file was created during the installation. You can create or edit a BLAST.OPT directly with an ASCII text-file editor such as MS-EDIT or the BLAST editor (for details, see “BLAST.OPT Format” below).

The BLAST.OPT Search Path

BLAST, BHOST, and SETBHOST search for BLAST.OPT as follows:

- ◇ First, in the directory where BLAST.EXE or SETBHOST is stored.
- ◇ Second, in the directory specified by the OPTDIR environment variable (if it has been defined).
- ◇ Third, in the current directory (only if OPTDIR has *not* been defined).

Reading Multiple BLAST.OPT Files

When BLAST finds more than one BLAST.OPT file in your search path, most entries are overwritten by the last BLAST.OPT read. However, unique COMMPORT and NETSERVICE assignments in each BLAST.OPT are added to the list rather than overwritten. (Changes to existing COMMPORT entries—such as specifying a new driver ID—are still overwritten.) This handling of BLAST.OPT files allows you to selectively install LAN drivers at run time based on your BLAST.OPT search path in order to optimize memory usage when BLAST is run in background mode.

BLAST.OPT Format

BLAST.OPT settings may be entered in any order using the following format:

SETTING=VALUE

To turn the 16550 option on, for example, use the assignment:

16550=ON

You may enter as many of the options as you need; spaces on either side of the equals sign are optional.

BLAST.OPT Settings

Following is a description of possible BLAST.OPT options and possible values for each option.

16550 **ON OFF**

Enables the FIFO buffers of the 16550 UART, a chip that resides on the serial cards of some PCs and internal modem. Enabling the FIFO buffers may improve throughput; however, some early versions of the 16550 chip are unreliable at speeds below 9600 bits per second. The default, which is set during the running of BINSTALL, is hardware dependent.

ACTFILE **[BLASTDIR\BHOST.ACT]**
(BHOST Only) **any valid path and filename**

Specifies an alternate BHOST accounts file.

BANNERTIME **0 – 99 [4]**

Specifies the number of seconds the BLAST banner will display on the screen before moving to the Offline menu. A setting of 0 causes the banner to be flashed and immediately removed.

EXAMPLE:

BANNERTIME=1

BG_BLK_SIZ **1 – 2048 [200]**
(Background Mode Only)

Specifies the maximum allowable block size, in bytes, for BLAST transfers in background mode. Like the MEMPOOL setting, a lower BG_BLK_SIZ setting reduces the amount of memory required for

background operation. If the Packet Size setup field is greater than this value, BLAST will automatically change it to BG_BLK_SIZ.

The default for this field, 200, should provide optimum performance for most situations (and is the required setting for transfers with BHOST). A setting of 84 saves memory but decreases throughput. Set BG_BLK_SIZ to 1 only when you will not perform any file transfers during a session.

EXAMPLE:

```
BG_BLK_SIZ=000
```

BLASTDIR any valid path

Specifies the directory where BLAST and BHOST look for BLAST.HLP, MODEMS.SCR, and SYSTEMS.SCR. BHOST additionally looks for its accounts file and log file in this directory.

This value is written to the BLAST and BHOST executables during installation. It may be changed through BLAST.OPT or by setting the DOS environment variable BLASTDIR to the desired location.

Note that the BHOST accounts file and log file may also be independently specified through ACTFILE and LOGFILE BLAST.OPT assignments.

EXAMPLE:

```
BLASTDIR=C:\BLAST
```

BREAKLEN 0 – 99 [10]

Specifies (in hundredths of a second) the duration value for the break character.

EXAMPLE:

```
BREAKLEN=50
```

COMMPORT label,irq,string_value

Specifies your communications port or network driver that will appear in the Connection setup field. *Label* is the name of the port or network driver; *irq* is a hexadecimal number specifying an interrupt request line or a command for loading a driver; and

string_value is dependent on the value of *irq* as shown below:

<i>irq</i>	<i>string_value</i>
1–F (specifies hardware interrupt)	port's hardware base address (in hexadecimal)
10–7F (specifies software interrupt)	optional text description
80–BF (specifies loading of driver)	load command for specific BLAST driver

If BLAST does not recognize *label*, *label* becomes a user-defined network service name.

Possible settings and examples are as follows:

Asynchronous Ports

Label is the name of the port, *irq* is the port's interrupt request line (hexadecimal 1–7F), and *string_value* is the port's base address.

EXAMPLE:

```
COMMPORT=COM4: , 0B, 2E8
```

specifies that COM4: use IRQ 11 and a base address of 2E8.

The communications port default values are:

Port	IRQ	Base Address
COM1:	4	3F8
COM2:	3	2F8
COM3:	4	3E8
COM4:	3	2E8
COM5:	3	4220
COM6:	3	4228
COM7:	3	5220
COM8:	3	5228

IMPORTANT: PS/2s and some PC compatibles use the following defaults:

Port	IRQ	Base Address
COM3:	3	3220
COM4:	3	3228

Extended BIOS (INT 14 Connections)

Label is the name of the port, *irq* is the label's interrupt request line (hexadecimal 10–7F), and *string_value* is an optional description.

EXAMPLE:

```
COMMPORT=COM5:,14,redirected
```

specifies that COM5: be patched to interrupt request line 14.

Network Connections Using BLAST Drivers

Label is the name of the network driver, *irq* is a hexadecimal number (80–BF) specifying that a BLAST driver is to be loaded, and *string_value* is the load command for a specific BLAST driver.

EXAMPLE:

```
COMMPORT=TCP/IP,85,blasttcp.exe
```

specifies that the BLAST TCP/IP driver be loaded by running the BLASTTCP executable.

IMPORTANT: There are restrictions to the use of LAN drivers under Windows 9x/NT. For help in setting up a LAN connection under these systems, consult our technical support staff.

COMPBUF (BHOST Only)

[0] – 50000

Allocates memory for data compression buffers. BHOST can take advantage of up to 50,000 bytes to speed file transfers.

BHOST uses data compression during file transfers even if this field is set to 0; to use compression level 4, however, you must set COMPBUF to at least 25000.

Slow PCs running at high baud rates or PCs performing node-to-node file transfers may operate less efficiently with the added overhead of compression. See “Compression Levels” on page 121 for more information on data compression.

EXAMPLE:

```
COMPBUF=25000
```

EDITOR

[BLASTEDT.EXE]
path\name /arguments

Specifies an alternate text editor BLAST will call when using the Local menu Edit command. If the program is located in the current working directory or contained in your path, you may specify just the editor name (with or without an extension). If the editor program is located elsewhere, you should include the entire path. Optionally, arguments may be passed to the editor with switches in the form of */command_line_switch*.

EXAMPLE:

```
EDITOR=C:\BIN\PROEDIT.EXE /b
```

FILEBUF 2048 – 65536 **[20480]** (BLAST)
 4000 – 65536 **[4000 or 7680]** (BHOST)

Specifies the number of bytes to be used as file buffers during file transfers. Large buffers improve throughput by allowing more of the file to be read at one time, thus minimizing disk access. Small buffers allow BHOST and BLAST (in background mode) to use less memory, leaving more memory available for foreground applications.

The default and minimum values for BHOST are 4000 (for MDA, CGA, and Hercules systems) and 7680 (for EGA, MCGA, and VGA systems).

NOTE: If you need to specify different values for BLAST and BHOST, use separate BLAST.OPT files and place them in separate directories.

EXAMPLE:

```
FILEBUF=32768
```

GROUND
(Background Mode Only)

[BACKGROUND]
 FOREGROUND

Specifies which screen the PC will display when BLAST is started in background mode. If no setting is specified or if BACKGROUND is specified, BLAST starts in interactive mode. If FOREGROUND is specified, BLAST starts in no-display mode and immediately returns control to the DOS system prompt. You may toggle between BLAST and the system prompt (or a program started from the system prompt) using the ALT H Hot Key.

EXAMPLE:

GROUND=FOREGROUND

LOGFILE
(BHOST Only)

[BLASTDIR\BHOST.LOG]
any valid path and filename

Specifies an alternate session log file.

EXAMPLE:

LOGFILE=C:\TEMP\DAILY.TXT

MEMPOOL
(Background Mode Only)

6836 – 128000 **[90000]**

Specifies the amount of memory BLAST will allocate for background operation before installing itself as memory-resident. This memory pool is used by BLAST for saving screens, creating windows, buffer compression, and emulator control. PCs with smaller amounts of available RAM may require a smaller MEMPOOL for BLAST to operate in background mode.

EXAMPLE:

MEMPOOL=8192

NETSERVICE

service name, label

Specifies a network service name or destination name to be added to the list of connection names available through the Connection setup field.

IMPORTANT: *Label* must have been previously defined as a network connection using a COMMPORT setting.

EXAMPLE:

NETSERVICE=zeus(196.85.214.57),TCP/IP

PRINTERCHK

NO [YES]

Enables/disables BLAST to perform checks on the printer (to see that it is online and not out of paper). Set PRINTERCHK to NO to re-direct PRN: printer output to a non-printer device using the MODE command. For example:

C:> MODE LPT1:=COM2:

will redirect line printer 1 output to the COM2: port.

EXAMPLE:

PRINTERCHK=NO

SETUPDIR

any valid path

Specifies the directory where BLAST and BHOST look for setup files. This value is written to the BLAST and BHOST executables during installation. It may be changed through BLAST.OPT or by setting the DOS environment variable SETUPDIR to the desired location.

EXAMPLE:

SETUPDIR=C:\JOHN\BLAST

SLICE

0 – 9999 [**30 or 100**]

Sets the amount of time BLAST passes control to the foreground task when started in Background mode or when run in a multitasking environment such as Windows and Desquview. The value entered here represents the number of times BLAST will execute its event loop before passing the PC's processor to the foreground task. For example, SLICE=99 will result in BLAST passing control to the foreground task after every 99 event checks. Note that a value of 0 is equivalent to a value of 9999. The default for 80286 CPU and faster machines is 100; the default for slower machines is 30.

EXAMPLE:

SLICE=500

SPEAKER (BHOST Only)

OFF

Disables the Host PC's internal speaker. By default, BHOST allows the foreground application to control the speaker. When this field is set to OFF, BHOST turns the speaker off each time it scans the application screen. If the application turns the speaker on, the speaker will remain on until the next time BHOST scans the application screen. OFF is the only acceptable value; otherwise this feature is disabled.

EXAMPLE:

SPEAKER=OFF

TCPINGW1AD

valid IP address

Specifies the network gateway address of a router on the network, in dotted.decimal form.

EXAMPLE:

TCPINGW1AD=128.1.17.254

TCPINLOCAD

valid IP address

Specifies your PC's dotted.decimal internet protocol address.

EXAMPLE:

TCPINLOCAD=128.1.17.44

TCPINSNMSK

valid IP address

Specifies the network mask address, in dotted.decimal form.

EXAMPLE:

TCPINSNMSK=255.255.255.0

TTRAP LEN

2 – [10]

Specifies the number of incoming characters Learn Mode will trap in a TTRAP statement.

EXAMPLE:

TTRAP LEN=4

VIDEOBUF

DISK

(Background Mode Only)

Specifies that the foreground screen be stored to disk rather than in memory when the ALT H Hot Key is used, thus saving memory. It is important that you have sufficient free disk space to use this function (4K for monochrome operation, 16K for CGA, and up to 256K for EGA/VGA). DISK is the only valid setting; if no setting is specified, the foreground screen is saved in memory.

EXAMPLE:

VIDEOBUF=DISK

VIDEOMODE

hexadecimal video mode

Specifies your EGA or VGA video adapter's 132 column by 25 line mode setting in hexadecimal for use with the VT and WYSE emulator 132 Compressed function. If you have selected 132 Compressed, BLAST will automatically attempt to set your video mode. If BLAST does not recognize your video adapter, you will need to specify your correct video mode with this setting. The actual value of this field varies from manufacturer to manufacturer; check your adapter card's technical reference manual or contact the manufacturer for the correct mode number. Your video adapter must use the standard DOS "Set Video Mode" function.

EXAMPLE:

```
VIDEOMODE=2B
```

Using BLAST on a LAN

In order for computers to communicate with each other over a LAN, they must be connected both physically, through cables and LAN adapters (such as Ethernet), and logically, with LAN software (such as Novell Netware).

At the heart of the LAN software is a network protocol that tells the computers how to send and recognize messages from other computers on the network. Novell Netware, for example, uses the IPX protocol, while the IBM-PC Network uses the NETBIOS protocol. In addition, special protocols are required to communicate with devices that are not directly connected to the network, including modems.

For maximum compatibility, BLAST is capable of communicating with a number of different LAN protocols. These include:

ACS	IBM's Asynchronous Communications Server
NMP	Network Products' communication server
IPX	Novell Netware
NETBIOS	IBM PC Net and others.
NETBIOSC	NetBIOS character mode
TCP/IP	Transfer Control Protocol/Internet Protocol

BLAST Network Drivers

In order to communicate with each kind of network protocol, BLAST must load a driver that supports that network protocol. The

BLAST drivers and corresponding executables for them are as follows:

<u>BLAST Driver</u>	<u>Executable</u>
ACS	BLSTACS.EXE
NMP	BLSTNMP.EXE
NETBIOS	BLSTLAN.EXE
NETBIOSC	BLSTLAN.EXE
IPX	BLSTIPX.EXE
TCP/IP	BLSTTCP.EXE

If a driver is specified during BLAST installation, it is given an assignment in a BLAST.OPT file. If it has a BLAST.OPT assignment and is specified in the Connection setup field, the driver is loaded automatically when the user enters the Online menu. Below are descriptions of BLAST drivers and options for loading the drivers from the command line. For a discussion of loading BLAST drivers when using background mode, see “Considerations for BLAST Configured for LANs” on page 17.

IMPORTANT: In the following command line descriptions, the underlined space (“_”) is required between the switch and the value following it.

ACS Driver

This driver provides access to IBM compatible ACS servers and supports modem sharing, including outbound and inbound calls. The driver occupies approximately 16.5K of RAM and is loaded when selected in a setup if it has been assigned in a BLAST.OPT file. It may also be loaded manually by typing:

```
blastacs /q /i_netid /b_bsize /x_brkcnt /s_paksiz /c_ictimer /p_ptimer /l /k
```

where:

/q

suppresses initial driver messages when loading the driver. If used, it must be the first switch.

/i_netid

80 – 9F [81]

specifies the network ID in hexadecimal.

/b_bsize 84 – 4096 [132]

specifies the network buffer size in bytes.

/x_brkcnt [1]

specifies the break signal count.

/s_paksiz 1 – 1024 [128]

specifies the maximum size of a packet transferred to the LAN device from the ACS server.

/c_ictimer 1 – 1024 [4]

specifies, in 20 millisecond units, the maximum time between the arrival of two characters. When this amount of time expires between the arrival of two characters, the buffered data in the server is sent to the LAN device. The default value, 4, is a good setting for terminal emulation. To optimize file transfer, increase this number.

/p_ptimer 1 – 1024 [25]

specifies, in 20 millisecond units, the amount of time the server waits until a packet is full before the buffered data is transferred to the LAN device.

/1

specifies that lana 1 should be used rather than the default, lana 0.

/kis

unloads the driver and reclaims RAM used by the driver.

NMP Driver

In conjunction with an NMP NCSI or NMP NASI interface, this driver provides modem sharing on both NETBIOS and Novell LANs. Therefore, whether the underlying transport layer is IPX or NETBIOS is of no consequence to the driver. However, the proper NCSI or NASI network driver must be loaded into RAM before you start BLAST.

The BLAST NMP driver interfaces to the NCSI or NASI extended functions—the driver does not have to use the NCSI or NASI Command Interpreter to connect to the NMP server software. The BLAST NMP driver occupies approximately 10K of RAM.

The driver is loaded when selected in a setup if it has been assigned in a BLAST.OPT file. It may also be loaded manually by typing:

```
blastnmp /q /i_netid /b_bsize /1 /k
```

where:

/q

suppresses initial driver messages when loading the driver. If used, it must be the first switch.

/i_netid 80 – 9F [82]

specifies the network ID in hexadecimal.

/b_bsize 84 – 4096 [128]

specifies the network buffer size in bytes.

/1

specifies that lana 1 should be used rather than the default, lana 0.

/k

unloads the driver and reclaims RAM used by the driver.

IPX Driver

This driver requires that the Novell IPX driver be loaded. It is not necessary to have logged onto a server.

The BLAST IPX driver is loaded when selected in a setup if it has been assigned in a BLAST.OPT file. It occupies approximately 26K of RAM. It may also be loaded manually by typing:

```
blastipx /q /i_netid /b_bufsize /s_sndbufs /r_rcvbufs /k
```

where:

/q

suppresses initial driver messages when loading the driver. If used, it must be the first switch.

/i_netid 80 – 9F [A3]

specifies the network ID in hexadecimal. A network ID of A0 or greater signifies block mode.

/b_bufsize 1 – [545]

specifies the network buffer size.

/s_sndbufs 1 – [2]

specifies the number of send buffers.

/r_rcvbufs 1 – [18]

specifies the number of receive buffers.

/k

unloads the driver and reclaims RAM used by the drive.

NETBIOS Driver

This driver provides access to login servers, LAN terminal emulation, and BLAST file transfer. The driver occupies approximately 11.5K of RAM and is loaded when selected in a setup if it has been assigned in a BLAST.OPT file. It may also be loaded manually by typing;

```
blastlan /q /i_netid /b_bsize /l /k
```

where:

/q

suppresses initial driver messages when loading the driver. If used, it must be the first switch.

/i_netid

80 – BF [A0]

specifies the network ID in hexadecimal. A network ID of A0 or greater signifies block mode.

/b_bsize

84 – 4096 [128]

specifies the network buffer size in bytes.

/l

specifies that lana 1 should be used rather than the default, lana 0.

/k

unloads the driver and reclaims RAM used by the drive.

TCP/IP Driver

This driver supports telnet (port 23) and FTP connections. It requires that a Clarkson compatible packet driver be loaded. Most network interface card vendors supply drivers that adhere to the Packet Driver specification.

The BLAST TCP/IP driver occupies approximately 88K of memory and is loaded when selected in a setup if it has been assigned in a BLAST.OPT file and configured through the BLAST.OPT options TCPINLOCAD, TCPINGWLAD, and TCPINSNMSK (for details on these options, see pages 27–27). Following are examples of the installation details for setting up BLAST to use TCP/IP on a PC running ODI. This information is provided as a working example only.

Fragment of the BLAST.OPT file:

```
COMMPORT=TCP/IP,85,BLASTTCP.EXE
TCPINLOCAD=198.85.116.29
NETSERVICE=blaster(198.85.116.11),TCP/IP
```

Fragment of NET.CFG:

```
Link Driver NE2000
  PORT 320
  INT 10
  FRAME Ethernet_II
  FRAME Ethernet_802.3
```

Fragment of CONFIG.SYS:

```
DEVICE=C:\DOS\SETVER.EXE
DEVICE=C:\DOS\HIMEM.SYS
DOS=HIGH
FILES=40
STACKS=9,256
BUFFERS=30
```

Fragment of AUTOEXEC.BAT:

```
C:\DOS\SMARTDRV.EXE /X
ECHO ON
PROMPT $p$g
PATH C:\NWCLIENT;C:\DOS;C:\BLAST
SET SETUPDIR=C:\BLAST
SET BLASTDIR=C:\BLAST
CD C:\NWCLIENT
LSL
NE2000
ODIPKT
```

Fragment of the screen output when the PC is booted:

```
C:\NWCLIENT>LSL
NetWare Link Support Layer v2.01 (921105)
(C) Copyright 1990, 1992 Novell, Inc. All
Rights Reserved.
```

```
Max Boards 4, Max Stacks 4
```

```
C:\NWCLIENT>NE2000
Novell NE2000 Ethernet MLID v1.34 (910603)
(C) Copyright 1991 Novell, Inc. All Rights
?Reserved.
```

```
Int 10, Port 320, Node Address 8060000085
Max Frame 1514 bytes, Line Speed 10 Mbps
Board 1, Frame ETHERNET_II
Board 2, Frame ETHERNET_802.3
```

```
C:\NWCLIENT>ODIPKT
ODIPKT 1.1
(c) Copyright Daniel Lanciani 1991-1992. All
rights reserved.
This software is provided with NO WARRANTY.
Using Ethernet framing, class 1
ODIPKT is installed and ready.
```

Using the BLAST Editor in LAN Environments

When you specify a filename to edit with the BLAST editor, always prefix the filename with a full path. If the file is in the current directory, use the prefix “\.” (period backslash) to insure that the file is saved to the correct directory.

Flow Control

Flow control paces the data stream between computers in order to prevent the receiving computer from losing characters. Flow control can be used by the receiving computer when:

- ◇ Its buffers are full and need to be written to disk, displayed, or printed before continuing.
- ◇ System resources are in heavy demand.
- ◇ Disk operations, such as file opening and writing, require time to complete.
- ◇ The communications interpreter is overloaded with requests from multiple devices.
- ◇ The receiving modem is error-controlled and its buffers are full.

RTS/CTS Pacing

The RTS/CTS Pacing setup field and @RTSCTS reserved variable can enable a form of flow control using the RS-232 signals Request-to-Send and Clear-to-Send. It is sometimes referred to as “hardware” or “out-of-band” flow control. The valid settings are YES to enable hardware flow control and NO to disable it. Unless you have an error-correcting modem (such as V.42 or MNP), you will not be able to use RTS/CTS flow control.

When the RTS/CTS Pacing setup field or @RTSCTS is set to YES and the device attached to your communications port (such as a modem, printer, or multiplexor) holds CTS low, BLAST will not send data. Consequently the communications process may appear to hang. Conversely, if your communications port holds RTS low, a device set up to sense RTS will appear to hang and will lose characters sent to it.

BLAST can control RTS in scripts with the RAISE RTS and DROP RTS statements.

XON/XOFF

The XON/XOFF Pacing setup field and @XONXOFF reserved variable allow BLAST to use “software” or “in-band” flow control during terminal and BLAST transfer operation. XON/XOFF flow control allows two computers to control the flow of data between them. When one computer needs to stop the flow of incoming data, it transmits XOFF (CTRL S) to the other computer. When the computer is again ready to receive data, it transmits XON (CTRL Q).

XON/XOFF is the most widely used form of flow control. However, it has potential problems:

- ◇ The protocol must not use the flow control characters to carry data. A good example is the Xmodem protocol, which can never be used through a device that honors XON/XOFF control.
- ◇ Both ends must implement a procedure to restart transmission if the XON character is lost or the transmission will be irrevocably halted. BLAST, for example, automatically resends a packet if it does not receive an XON within 30 seconds of receiving an XOFF.
- ◇ The sequence of flow control can get very cumbersome when the computer, a modem, a network, and a terminal server are all exerting flow control on each other with XON/XOFF.

Chapter 3

BLAST Quickstart

IMPORTANT: The following section assumes that BLAST has been properly installed. Before proceeding, be sure to:

- ◇ Successfully complete the entire BLAST installation process as instructed in the BLAST Installation Guide.
- ◇ Connect the modem according to the instructions supplied by the modem manufacturer and turn on the modem.

Starting BLAST

The command to execute BLAST is issued at the operating system prompt. Type:

```
BLAST
```

and press the ENTER key.

If BLAST does not start up, make sure that you are in the directory where the BLAST program is located or that you have added the BLAST directory to your PATH. Consult the documentation that came with your version of DOS to learn how to modify the PATH.

If this is the first time you have run BLAST, the Online Help screen appears automatically (only on the first time). You can either explore the Help menu now or press ESC to continue. Otherwise, BLAST displays the Offline menu. At this point, you can control BLAST interactively.

The BLAST Screen

The BLAST screen (Figure 3-1) includes three sections: the Command Area, the Scrolling Region, and the Status Line.

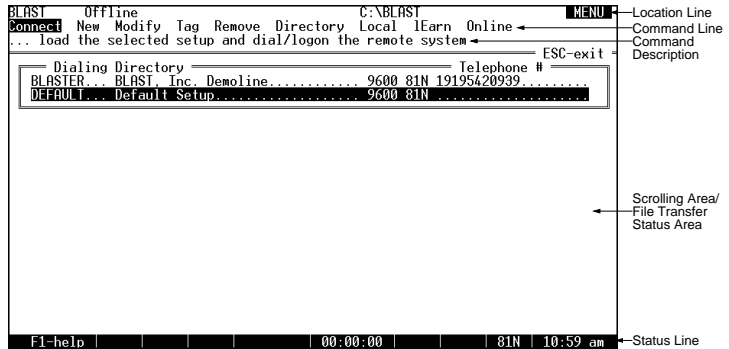


FIGURE 3-1

Command Area

The Command Area consists of three lines: the Location Line, the Command Line, and the Command Description.

Location Line

The Location Line (Figure 3-2) provides information about your location within BLAST. The items in the Location Line are:

Current Menu – displays the BLAST menu currently in use. The possible values are Offline, Online, Filetransfer, Local, and Remote.

Active Setup – displays the setup that is currently loaded (not displayed in the Offline menu).

Current Directory – identifies the working disk directory. Use the Chdir command in the Local menu to change the current directory.

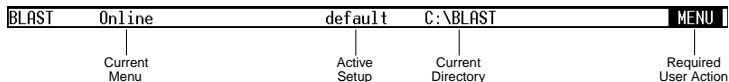


FIGURE 3-2

Required User Action – displays the action that BLAST expects from you. Possible values are:

- MENU – select a command from the menu.
- INPUT – type in data at the prompt.
- ERROR – review the error message, then press any key.
- WAIT – no action allowed, BLAST is busy.
- SCRIPT – a BLAST script is executing.
- ONLINE – BLAST is online.

Command Line

The Command Line lists all the commands available for the active menu.

Command Description

The Command Description gives a one-line explanation of the command currently highlighted by the cursor. If you need more information about the command, press the F1 key for Online Help.

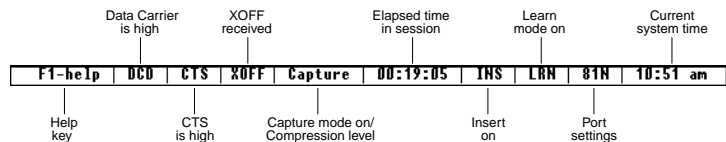
Scrolling Region

The Scrolling Region is the area below the Command Area. If you are in the Offline menu, the Dialing Directory is displayed in this area. When performing a function, BLAST uses this area to display status and data. The information displays in a variety of formats, depending on the activity. This is also the File Transfer Status Area, described below.

Status Line

The Status Line (Figure 3-3) displays the status of the current communications session. Each indicator has a fixed position in the Status Line. If the condition is true, the indicator will be displayed. For example, if DCD appears on the Status Line in the second position, it indicates the existence of a data carrier.

FIGURE 3-3



File Transfer Status Area

The File Transfer Status Area appears only during a file transfer session between two systems. It displays information concerning the

files being sent and/or received. Figure 3-4 shows the status informational displayed during a BLAST protocol file transfer session. Other filetransfer protocols supported by BLAST have displays tailored to the capabilities of the protocol.

```

BLAST Filetransfer default C:\BLAST MENU
Send Get Message Remote Local File
... send file(s) to the remote system
== local == opt = % xfer = file size = byte cnt = ln qual ==
S: <idle> good (00)
R: <message> ESC-exit
-----<< Entering BLAST Transfer Mode >>-----
**** MacBLAST 10.2 on remote system [uovm] ****

F1-help | DCD | CTS | | | 00:00:59 | | | 81N | 11:11 am

```

FIGURE 3-4

Following is a description of each item, or status indicator, in the BLAST protocol File Transfer Status Area.

local – the name of the file that your system is sending or receiving.

opt – the optional transfer switches that you selected for this file.

%xfer – the percentage of the file that has been transferred to or from the remote machine.

file size – the total file size (in bytes).

byte count – the portion of the file that has been transferred to or from the remote machine (in bytes).

ln qual – a general description of the line quality of the connection between the computers. Possible values during a transfer are good, fair, poor, or dead.

Unlike BLAST protocol, other supported protocols do not make use of all the above status indicators.

Three Keys to Remember

A number of special keys are used within BLAST, but three are used frequently:

CTRL K	CTRL K is the default “Attention (<i>ATTN</i>) Key.” Press CTRL K to abort script operations or initiate other special key combinations. Press CTRL K CTRL K to return to the Online menu from Terminal mode. (For information on redefining the <i>ATTN</i> key, see “Attention Key” on page 72.)
ESC	Press ESC to cancel the current action, return to the previous menu, or exit BLAST.
F1	Press F1 for context-sensitive Online Help.

The BLAST Menus

Within menus, move from one command to the next by pressing the SPACEBAR or by using the cursor keys. Select a command by pressing the capitalized letter in the command or by pressing ENTER when the cursor rests on the desired command. After opening a submenu, return to the previous menu by pressing ESC.

Below the menu is a one-line description of the current command (Command Description Line). To get more information, press F1 when the cursor highlights the appropriate command. After displaying text related to the command, BLAST displays a general help section on topics not specifically related to any command. See Chapter 4 for a detailed discussion of the menus.

NOTE: Online Help is accessed by pressing F1 everywhere in BLAST except for Terminal mode, where it is accessed by pressing CTRL K H.

Menu Summary

Each of the menus offers commands that are grouped together by function. For example, the Local menu allows you to manage your system while online with a remote system, whereas the Filetransfer menu provides functions connected with sending and receiving files.

Following is a brief summary of each menu and its purpose:

Offline – Manages setups (New, Modify, Tag, Remove, and Directory); loads the selected setup and dials out (Connect); and starts Learn mode (LEarn).

Online – Manages connecting to and disconnecting from a remote system (Connect and Disconnect); launches a BLAST

script (Script); uploads and captures text files (Upload and caPture); controls a remote PC (Access); and starts Terminal mode (Terminal).

Filetransfer – Sends and receives files using either BLAST, FTP, Kermit, Xmodem, Ymodem, or Zmodem protocol.

Remote – Available with BLAST protocol and Kermit protocol. Performs file management tasks on the remote system, such as listing, renaming, deleting, typing, and printing remote files.

Local – Performs file management on your system (List, Delete, Rename, Type, Print, Chdir); calls the selected text editor (Edit); displays snapshots and movies (View); and provides access to the operating system command line (System).

A Quickstart File Transfer

The most common use of BLAST is communicating between two computers using standard asynchronous modems and ordinary telephone lines. BLAST provides hands-on experience in this environment by offering a dial-in computer system called Blaster available 24 hours a day, seven days a week for BLAST demonstrations and testing. You are encouraged to take advantage of this service to familiarize yourself with the many features of BLAST.

This section of Quickstart will guide you through:

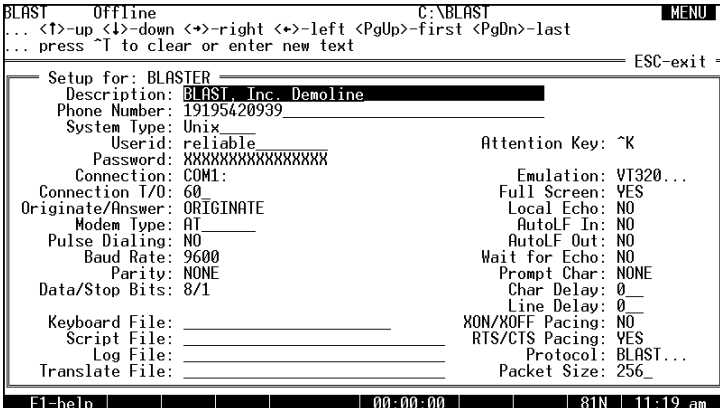
- ◇ Selecting the Blaster setup.
- ◇ Connecting to Blaster.
- ◇ Performing BLAST protocol transfers.
- ◇ Logging off Blaster.

Although we recommend that you complete this section in one sitting, you may elect to stop by returning to the Online menu and choosing the Disconnect command.

Selecting the Blaster Setup

Setups contain all the information that BLAST needs to connect to and communicate with remote computers. Each setup is a separate file created and modified through the Offline Menu. This process is described in detail in Chapter 5. For this demonstration, you will use a setup called Blaster, which was copied to your disk during BLAST installation.

If you have been moving through the menus, press the `ESC` key until you return to the Offline menu. You should see “Blaster” listed as one of the entries in the Dialing Directory. Use the cursor keys to highlight Blaster and then press `M`. You should see the Setup window shown in Figure 3-5 below.



```
BLAST Offline C:\BLAST MENU
... <T>-up <D>-down <R>-right <L>-left <PgUp>-first <PgDn>-last
... press ^T to clear or enter new text ESC-exit

Setup for: BLASTER
Description: BLAST, Inc. Demoline
Phone Number: 19195420939
System Type: Unix
Userid: reliable Attention Key: ^K
Password: XXXXXXXXXXXXXXXXXXXX
Connection: COM1 Emulation: VT320...
Connection T/O: 60 Full Screen: YES
Originate/Answer: ORIGINATE Local Echo: NO
Modem Type: AT AutoLF In: NO
Pulse Dialing: NO AutoLF Out: NO
Baud Rate: 9600 Wait for Echo: NO
Parity: NONE Prompt Char: NONE
Data/Stop Bits: 8/1 Char Delay: 0
Line Delay: 0
Keyboard File: XON/XOFF Pacing: NO
Script File: RTS/CTS Pacing: YES
Log File: Protocol: BLAST...
Translate File: Packet Size: 256

F1-help 00:00:00 81N 11:19 am
```

FIGURE 3-5

Check to see that the following entries appear correctly in the Setup window:

Phone Number:	1-919-542-0939
System Type:	UNIX
Userid:	reliable
Password:	XXXXXX (fast-transfer is the actual password, but it will be masked by “Xs”)
Parity:	None
Data/Stop Bits:	8/1
Emulation:	VT320
Protocol:	BLAST

If any of the entries are incorrect, use the cursor keys to move to the appropriate field and enter the information. For the fields Phone

Number, Userid, and Password, you will need to type in the correct information (the userid and password are case-sensitive and should be typed exactly as they appear above). For the remaining fields illustrated above, you can cycle through the available choices by pressing either the SPACE or the BACKSPACE key. After you are satisfied that all of the setup information is correct, press ESC to exit to the Offline menu. If you made any changes to the setup, you will be asked if you want to save changes and will be prompted to answer “Yes” or “No”—press Y to save the changes and return to the Offline menu.

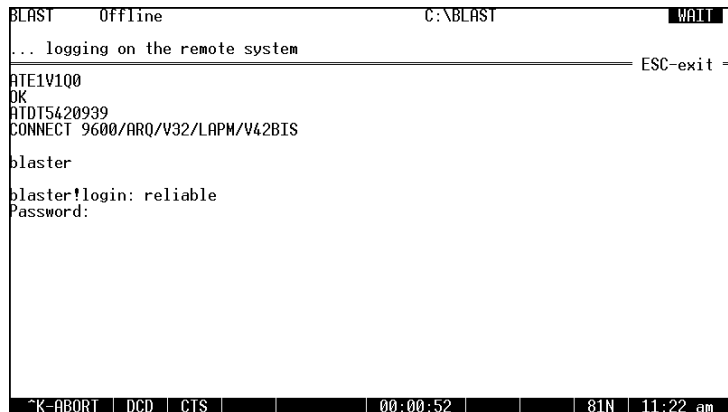
Connecting to Blaster

Your PC is now ready to begin talking to Blaster. You will connect to Blaster from the Offline menu by using the Connect command, which will automatically dial Blaster.

The screen will display messages for each of the steps in the Connect process. If your modem has a speaker, listen to make sure that it dials the number. Also, watch the terminal dialogue between the computer and the modem. When the call is successful, a message will display indicating that the connection has been established:

```
CONNECT nnnn
```

where *nnnn*, if present, gives additional information about the quality and speed of the connection (Figure 3-6).



```
BLAST Offline C:\BLAST WAIT
... logging on the remote system
----- ESC-exit
ATE1V1Q0
OK
ATDT5420939
CONNECT 9600/ARQ/V32/LAPM/V42BIS
blaster
blaster!login: reliable
Password:
^K-ABORT | DCD | CTS | | 00:00:52 | | 81N | 11:22 am
```

FIGURE 3-6

After recognizing the modem’s CONNECT message, Blaster’s banner and request for login will be displayed. Your setup file will automatically enter the userid and password. When the login is

complete, BLAST returns control to you by displaying the Online menu (Figure 3-7) and waiting for your input.

```
BLAST Online BLASTER C:\BLAST MENU
Connect Terminal capture Upload Filetransfer Script Local Access Disc
... become a terminal to the remote system ESC-exit

WELCOME to BLAST, INC.

You have logged into Blaster, our SCO Unix machine at BLAST, Inc.
This login is for the use and convenience of customers of BLAST, Inc.
Under this login, you can get sample scripts from the scripts
directory and technical notes from the text_documents directory.

$

F1-help DCD CTS 00:01:14 81N 11:22 am
```

FIGURE 3-7

Performing BLAST Protocol Transfers

To perform a BLAST protocol transfer, first select the Filetransfer command from the Online menu (Figure 3-7) by pressing F. In a moment, Blaster will synchronize with your system and the Filetransfer menu (Figure 3-8, next page) will be displayed.

Getting a File from Blaster

To get a file from Blaster:

◇ Select Get by pressing G.

◇ At the prompt:

enter remote filename:

type:

blaster.msg ENTER

◇ At the prompt:

enter local filename:

type:

news.msg ENTER

◇ At the prompt:

specify transfer options (t=text, o=overwrite, a=append):

type:

t ENTER

to transfer the file using text format translation.

BLAST will begin retrieving the file, and the byte count in the File Transfer Status Area will increase. After the file has been completely sent, the byte count will stop, a blank will appear in the byte count status indicator, and the following message will be displayed at the bottom of the screen as shown in Figure 3-8 below:

news.msg/T=TXT ... receive completed

```
BLAST Filetransfer BLASTER C:\BLAST MENU
Send Get Message Remote Local File
... get file(s) from the remote system
-- local opt = % xfer = file size = byte cnt = ln qual =
S: <idle> good
R: <idle> good (00)
ESC-exit -
You have logged into Blaster, our SCO Unix machine at BLAST, Inc.
This login is for the use and convenience of customers of BLAST, Inc.
Under this login, you can get sample scripts from the scripts
directory and technical notes from the text_documents directory.
$
$ blast -h
starting BLAST protocol
-----<< Entering BLAST Transfer Mode >>-----
**** BLAST Professional UNIX 10.7.5n1 on remote system [uov] ****
news.msg/T=TXT ... receive completed
F1-help DCD CTS SCO-RC4 00:03:27 81N 11:32 am
```

FIGURE 3-8

Sending a File

To send a file to Blaster:

◇ Select Send by pressing s.

◇ At the prompt:

enter local filename:

type:

news.msg ENTER

◇ At the prompt:

enter remote filename:

type:

news.msg ENTER

◇ At the prompt:

specify transfer options (t=text, o=overwrite, a=append):

type:

tO ENTER

to transfer the file using text format translation and to overwrite any existing versions of the file.

Again, notice that the status fields are updated as the file transfer progresses. At the end of the transfer, you will see the following line displayed on your screen:

news.msg/T=TXT news.msg/OVW/T=TXT ... send completed

After the file transfer is complete, press ESC to return to the Online menu. An orderly shutdown of the BLAST protocol will follow. After a few seconds, the Online menu will be displayed.

Logging Off Blaster

To log off Blaster, select the Disconnect command by pressing D. To quit BLAST, press ESC twice. BLAST will prompt with:

Do you really want to leave BLAST? NO YES

Type Y to quit.

Chapter 4

The Menus

Moving Through the Menus

This chapter guides you through the various BLAST command menus. Some items are described in more detail in other chapters; in such cases, you will be referred to the appropriate chapter. A variety of special keys and support for a mouse allow you to navigate through the various commands quickly and easily. Each menu offers commands that are grouped together by function. For example, the Local menu allows you to manage your system online with a remote system, whereas the Filetransfer menu allows you to perform functions connected with sending and receiving files.

Within the command line of a menu, move from one command to another by pressing SPACEBAR, BACKSPACE or by using the left and right arrow (cursor) keys. Execute a command by pressing the capitalized letter in the command or by pressing ENTER when the cursor rests on the desired command. After opening a submenu, return to the previous menu by pressing ESC. For a discussion of selecting a setup and navigating through a Setup window, see “What is a Setup?” on page 61.

The Keyboard

BLAST uses special key sequences to differentiate between local commands and characters meant for the remote system. The BLAST Keys perform local functions, such as exiting Terminal mode, displaying Online Help, or starting the BLAST editor. BLAST Keys are most important in Terminal mode, when BLAST ordinarily sends all keystrokes directly to the remote computer. All of the BLAST keys are listed in Appendix E. Some of them can be reassigned using the BLAST keyboard utility, BLASTKBD (see “Keyboard Mapping Utility” on page 296 for details).

Three Keys to Remember

You will use three of the BLAST Keys most often:

- | | |
|--------|--|
| CTRL K | CTRL K is the default “Attention (<i>ATTN</i>) Key.” Press CTRL K to abort script operations; press CTRL K CTRL K to return to the Online menu from Terminal mode. |
| ESC | Press ESC to cancel the current action, return to the previous menu, or exit BLAST. |
| F1 | Press F1 for context-sensitive Online Help (In Terminal mode, press CTRL K H). |

The Attention Key

The Attention Key alerts BLAST to prepare for a particular operation. The Attention Key, represented in this documentation by the abbreviation *ATTN*, is actually two keys—CTRL plus another character. The default Attention Key is CTRL K. Press CTRL K (*ATTN*) to abort script operations or initiate other special key combinations. Press CTRL K CTRL K (*ATTN ATTN*) to return to the Online menu from Terminal mode.

You may change the default value of the Attention Key by altering the value of the Attention Key setup field (page 72) or by setting the BLASTscript reserved variable @ATTKEY (page 241).

NOTE: If it is necessary to change the Attention Key, be sure to choose a replacement value that will not interfere with your system’s designated control codes. In particular, do not use CTRL M, which is the control code for a carriage return. Check your system manual for more information about special control codes before you reassign the Attention Key.

The Attention Key can initiate many useful functions from Terminal mode. Please refer to Appendix B for all of the Attention Key sequences.

The Cancel Key

The ESC key is used to cancel the current action. It also returns you to a previous menu from a lower level menu and is used to exit BLAST from the Offline menu. The exception to this rule is that you must press *ATTN ATTN* to escape from Terminal mode.

The Help Key

F1 is the default context-sensitive Help key. When the cursor rests on a command in the menu, pressing F1 will display Help about that particular topic. After displaying text related to the command, BLAST displays a general help section on topics not specifically related to any command. Note that Help is the F1 key everywhere in BLAST except Terminal mode, where it is the *ATTN H* key sequence. The Help key can be reassigned using the BLAST keyboard utility, *BLASTKBD* (see Chapter 17 for details).

Other Special Keys

Two other special types of keys are available through BLAST—Hot Keys and Soft Keys (see Chapter 17 and Appendix B for details).

Hot Keys – Hot Keys access often-used functions from Terminal, Filetransfer, and Access modes. Typing *ALT F* from Terminal mode, for example, starts Filetransfer mode and automatically returns you to Terminal mode when the transfer is complete.

Soft Keys – Soft Keys allow you to send often-used character strings to a remote system with a single keystroke.

Using a Mouse

BLAST provides mouse support for a point-and-click selection of commands and easy maneuvering within the BLAST menus. To activate the mouse, load the mouse driver and start BLAST. You can then execute any command that appears on the screen by moving the mouse cursor over that command and clicking the left button.

In addition, BLAST supports several convenient mouse-only shortcuts:

- ◇ Clicking on `ESC-exit` in the right top portion of your screen is the same as pressing `ESC`.
- ◇ Clicking on `F1-help` in the status line displays Online Help.
- ◇ Clicking on a setup in the Dialing Directory will highlight it.
- ◇ Double-clicking on a setup name in the Dialing Directory loads that setup and dials for a connection.
- ◇ Clicking while in the Terminal window exits Terminal mode.
- ◇ Clicking on a setup input field in Modify mode advances the cursor to that field.
- ◇ Clicking on a command description at the top of the setup screen when a multiple-choice field is highlighted performs that command.
- ◇ Clicking on a multiple choice setup field when the keyboard cursor is already on that field displays the next option in the list.
- ◇ Clicking on the `XOFF` in the status line resets flow control if an `XOFF` has been received.
- ◇ Clicking on `Capture` in the status line toggles capture on or off (only after you have named a capture file).
- ◇ Clicking on the timer (`00:00:00`) in the status line resets the timer.
- ◇ Clicking on `LRN (Learn)` in the status line turns Learn mode off.

NOTE: Status line shortcuts are only available with the standard BLAST status line, not with the special VT320 or WYSE60 status lines.

The Offline Menu

The Offline menu (Figure 4-1, next page) is the first menu displayed when you execute the BLAST program. The display includes three sections: the Command Area, the Scrolling Area, and the Status Line. See “The BLAST Screen” on page 38 for a description of these sections. We will be concerned here primarily with the Command Area, specifically the Command Line.

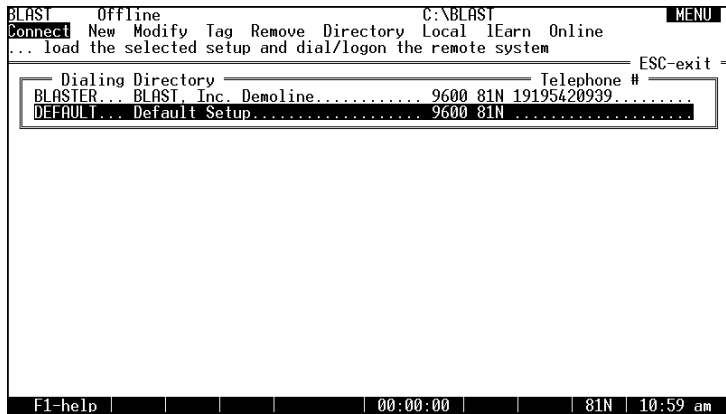


FIGURE 4-1

If this is the first time that BLAST has run, the help screen will appear. Press ESC to leave the Help screen. The Command Description Line below the Command Line offers a brief description of the highlighted command. To get more information, press F1 when the cursor rests on the appropriate command.

Setup Commands

Five of the commands in the Command Line of the Offline menu affect the setups listed in the Dialing Directory (see “What is a Setup?” on page 61 for more details). Following is a brief description of each command:

Connect – Loads the highlighted setup into memory and immediately attempts to dial the phone number contained in that setup. See “Modifying a Setup” on page 64 for details concerning the difference between this Connect command and the Connect command in the Online menu.

New – Prompts you for a new setup name. Type the name and press ENTER. BLAST will automatically enter the Modify mode and display in the Setup window the values of the default setup.

Modify – Displays the current values of the setup highlighted in the Dialing Directory. When Modify mode is exited, those values will be loaded into memory. If you made any changes, a prompt will ask whether or not you wish to save the changes.

Tag – Marks an individual setup for deletion.

Remove – Deletes setups that have been marked with the Tag command.

Other Offline Commands

Directory – Changes the current Dialing Directory (SETUPDIR— see Chapter 2 for details).

Local – Performs local system commands by taking you to the Local menu (described in detail on page 57).

Learn – Builds a script for you by starting Learn mode. When you execute the Learn command, you are prompted for a script name. After you type the name and press ENTER, BLAST records all of subsequent functions in the script file until you disable Learn mode by selecting the Learn command again. If you specify an existing filename for the script, BLAST asks whether you want to append to or overwrite the original script file. See “Learn Mode” on page 160 for more details.

Online – Takes you to the Online menu, described in the next section.

The Online Menu

Selecting Online from the Offline menu displays a menu like or similar to the one shown in Figure 4-2 (next page). All characters received and transmitted in Terminal, Capture, and Upload modes are filtered by the translate file if one is specified in the Translate File setup field (page 71). See “Translate File Format” on page 284 for more information on translate files. Following is a brief description of the commands of the Online menu:

Connect – Dials the phone number stored in memory from the current setup. See “Modifying a Setup” on page 64 for details about the difference between this Connect command and the Connect command in the Offline menu.

Terminal – Makes your system a terminal to the remote system. The menu commands will no longer be available to you. Remember that you must press *ATTN ATTN* in order to exit Terminal mode and return to the command menus. See “Terminal Emulation” on page 287 for more information.

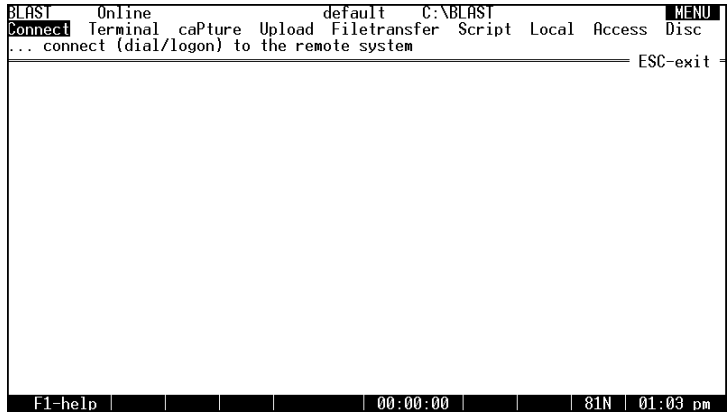


FIGURE 4-2

Capture – Causes all incoming text from the remote system to be captured to a file. When you enter Capture mode by executing the Capture command, you are prompted for a filename. After you type the name and press ENTER, BLAST records all of the subsequent text displayed in the Terminal window in the capture file until you disable Capture mode by selecting the Capture command again. If you specify an existing filename for the capture file, BLAST asks whether you want to append to or overwrite the original file. See “Downloading Text from a Remote Computer” on page 146 for more information.

Upload – Sends text from a local file to the remote computer and displays the text on your screen. When you execute the Upload command, you are prompted for a filename. After you type the name and press ENTER, BLAST displays the text on your terminal screen as well as sending it to the remote system. The remote system must have a text capture program in order to store the text in a remote file. See “Uploading Text to a Remote Computer” on page 145 for more information.

Filetransfer – Takes you to the Filetransfer menu described in the next section. See also chapters on individual protocols.

Script – Executes a BLAST script after prompting you to enter the script name. See Chapters 12–14 for information on scripts.

Local – Allows you to perform local system commands by taking you to the Local menu described on page 57.

Access – Begins a remote control session. After you enter Access mode, *ATTN* takes you to the Access menu (see “The Access Menu” on page 308).

Disc – Logs off the remote system cleanly and hangs up the modem using information from the System Type and Modem Type setup fields.

The Filetransfer Menu

Selecting the Filetransfer command from the Online menu takes you to the Filetransfer menu. The Filetransfer menu for BLAST protocol is shown in Figure 4-3 below. The commands on the Command Line of the Filetransfer menu vary depending on the protocol. For example, the Xmodem, Ymodem, and Zmodem protocols display only the Get and Send commands, whereas the Kermit protocol has additional options and its own special Remote submenu. Following is a brief description of the commands of the BLAST protocol Filetransfer menu. For more information on menu options for specific protocols, see chapters discussing individual protocols.

Send – Sends a file or files to the remote system.

Get – Retrieves a file or files from the remote system.

Message – Sends a message to the remote operator. Simply type the message and press ENTER. The message will be queued for transmission to the remote display.

```
BLAST Filetransfer default C:\BLAST MENU
Send Get Message Remote Local File
... send file(s) to the remote system
== local == opt = % xfer = file size = byte cnt = ln qual ==
$: <idle>
R: <message>
good (00)
ESC-exit
-----<< Entering BLAST Transfer Mode >>-----
**** MacBLAST 10.2 on remote system [uovm] ****

F1-help DCD CTS 00:00:59 81N 11:11 am
```

FIGURE 4-3

Remote – Performs remote system commands allowing limited access to the remote computer. The BLAST protocol Remote menu commands, which are similar to the Local commands, are described on page 120; see also “Kermit Remote Menu” on page 135.

Local – Performs local system commands. This command takes you to the Local menu, described in the next section. Note that all filetransfer activity is suspended while you are using the local system. This inactivity may exceed the interval specified by the BLAST protocol Inactivity Timeout setup field (page 88) and terminate Filetransfer mode.

File – Executes a transfer command file that can control an entire BLAST protocol transfer unattended (see “Transfer Command File” on page 117).

The Local Menu

The Local menu shown below in Figure 4-4 allows you to perform DOS commands on your local PC, including escaping to a DOS shell. Local commands affect only files in the current directory unless you specify a pathname.

```

BLAST Local HARDWARE C:\BLAST MENU
List Delete Edit Rename Type View Print Chdir System
... list filenames
-- local -- opt = % xfer = file size -- byte cnt -- ln qual --
S: <idle>
R: <message> good (00)
ESC-exit
TEST.3 6705 12-21-94 2:22p
HOLDING 37888 1-01-80 0:00a
LOG.SCR 754 12-30-94 3:54p
JAZZ.1 16052 12-30-94 9:35a
JAZZ.2 16052 12-30-94 9:35a
JAZZ.3 16052 12-30-94 9:35a
JAZZ.4 16052 12-30-94 9:35a
FTP.SO 6672 1-09-95 10:38a
BHOST.SO 6652 1-10-95 11:46a
VOHO 3956 1-09-95 8:40p
NEWS.MSG 22659 12-14-94 8:57a
VUYU 4478 1-18-95 5:17p
BLAST.TDF 16052 2-02-95 8:54a
XLOGGER.SU 6677 2-01-95 4:32p
B0013906 0 2-02-95 11:51a
79249408 bytes free

F1-help DCD CTS 02:46:18 81N 11:51 am
  
```

FIGURE 4-4

Following is a brief description of the commands of the Local menu.

List – Displays the contents of the current local directory. You will be prompted to choose either a detailed (long) or non-detailed (short) list and then to enter a filename; you may use a specific filename, a filename with wildcard characters (for exam-

ple, “*”), or press ENTER to display all files in the current local directory.

Delete – Erases a single file or multiple files. You may use a specific filename or a filename with wildcard characters (for example, “*”).

Edit – Invokes the BLAST editor (discussed in Chapter 11). You may also invoke a different editor by specifying it in a BLAST.OPT file (see the EDITOR BLAST.OPT option on page 24).

Rename – Renames a local file.

Type – Displays a local file in the scrolling area.

View – Displays either a “snapshot” or a “movie” that has been made using the Access menu (see “The Access Menu” on page 308).

Print – Prints a file to the local printer.

Chdir – Changes from the current local directory to one that you specify. The current directory is displayed on the top line of the BLAST screen. BLAST will check this directory for any files that you specify with the Local menu commands.

System – Performs a local system command. At the prompt, type a system command and press ENTER. Alternatively, you may simply press ENTER and escape to a system prompt that takes over the BLAST display. Typing EXIT and pressing ENTER returns you to BLAST.

NOTE: When BLAST is started with the /b switch (or with the /n switch if the display has not been re-enabled through a script), you cannot escape to a system prompt with this command (see “Command Line Switches” on page 10 and “BLAST in the Background” on page 13).

The Remote Menu

If you are using BLAST protocol or Kermit protocol, the Filetransfer menu contains a Remote command that takes you to the Remote menu. The Remote menu allows a user with no knowledge of the remote operating system to manage files on that system.

```

BLAST Remote HARDWARE C:\BLAST MENU
[Esc] Delete Rename Type Print Chdir More
... list remote filenames
== local == opt = % xfer = file size = byte cnt = ln qual ==
S: <idle> good
R: <message> good (00)
ESC-exit =
fig2-1.tif
fig2-2.tif
fig2-4.pic
fig2-4.tif
fig2-5.pic
fig2-5.tif
fig2-6.pic
fig2-6.tif
fig2-7.pic
fig2-7.tif
fig2-8.pic
fig2-8.tif
fig2-9.pic
fig2-9.tif
hwslave.scr
... use More (M) to resume display
F1-help DCD CTS 02:51:18 81N 11:56 am

```

FIGURE 4-5

Figure 4-5 above shows the BLAST protocol Remote menu. The commands of this menu, which differ from those of the Kermit Remote menu, are described briefly below. For a fuller discussion of the commands of the Remote menus, see “BLAST Protocol Remote Menu” on page 120 and “Kermit Remote Menu” on page 135.

List – Lists a remote directory.

Delete – Deletes a file one at a time from the remote system.

Rename – Renames a remote file.

Type – Displays a remote file on the BLAST screen.

Print – Prints a remote file to the remote printer.

Chdir – Changes the current remote directory.

More – Scrolls a page of data output from the List or Type commands.

Automation with BLASTscript

Up to this point, you have been learning about BLAST in interactive mode, manually pressing keys to perform tasks. To automate communications tasks that are repeated on a daily or weekly basis, use BLAST’s interpretive programming language, BLASTscript.

BLAST scripts can:

- ◇ Automate the dial and logon sequences to another computer.
- ◇ Send and receive files.
- ◇ Control standard and nonstandard modems and communication devices.
- ◇ Customize the user interface.
- ◇ Perform error-checking for session validation.
- ◇ Access online information services to send and receive mail.
- ◇ Poll large numbers of unattended remote sites after regular business hours.

Refer to Chapters 12–14 and Appendix E of this manual for detailed information on the use of BLAST scripts.

Chapter 5

The Setup

What is a Setup?

Communication between computers requires a great deal of information: the phone number of the remote computer, the modem type and baud rate, basic communications parameters, and more. BLAST keeps this information in individual files called “setups,” one file for each different system connection. BLAST is distributed with `BLASTER.SU`, a setup that contains the correct settings for you to call the BLAST demonstration line (see “Connecting to Blaster” on page 44). A setup containing default values, `DEFAULT.SU`, is created when BLAST is executed for the first time.

You can customize the setup by selecting the Modify command in the Offline menu. Although this chapter tells you how to create, edit, and save setups, the Online Help for some setup fields has more specific information.

We recommend that you make any changes to the setup through the Modify menu; however, setups are text files and can thus be edited with any text editor. Be sure to save the file as “text only” or “ASCII” and give it the extension “.SU”; do *not* save it as a word processor file.

Loading a Setup

In the Offline menu, you will see a Dialing Directory of all available setup files (Figure 5-1). There are three ways to load a setup file into memory:

1. To load the setup and connect in one step—use the up and down arrow keys to highlight the setup file in the directory and then press **C**. This both loads the setup file and attempts to connect to the number specified in the setup file. BLAST will automatically move to the Online menu.
2. To load the setup for possible modification—use the up and down arrow keys to highlight the setup file in the directory and then press **M**. This will load the setup values into memory and display the Setup window (Figure 5-2 on the next page). If you escape from this screen without making any modifications, the setup values will remain in memory. If you make changes to the setup, BLAST will prompt you to save or discard the changes. You will then be returned to the Offline menu.
3. To load the setup without connecting—press **S**; when you see the prompt, “Enter setup name to load,” type in the name of the setup file and press **ENTER**. BLAST will automatically move to the Online menu after loading the setup.

NOTE: If you load a mouse driver, you may perform these procedures with a mouse. For details on using a mouse, see “Using a Mouse” on page 51.

Notice that the name of the setup is now shown on the top line of the Online menu.

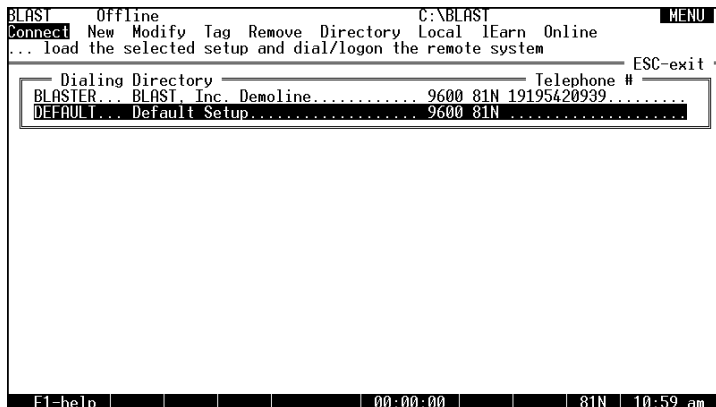


FIGURE 5-1

The Default Setup

BLAST creates a setup named DEFAULT.SU as part of the installation procedure. This setup contains default values for each setup field and is automatically loaded when you start BLAST (unless you specify another setup on the command line; see “Command Line Switches” on page 10). BLAST uses the default setup as the template for new setups. Each setup you create will use these same values until you modify them. By customizing DEFAULT.SU with the values you are most likely to need, you can make creating new setups faster and easier.

If you unintentionally overwrite the original DEFAULT.SU, you can restore its original settings by deleting or renaming the existing DEFAULT.SU and restarting BLAST. BLAST will create a new DEFAULT.SU based on the values you entered during installation.

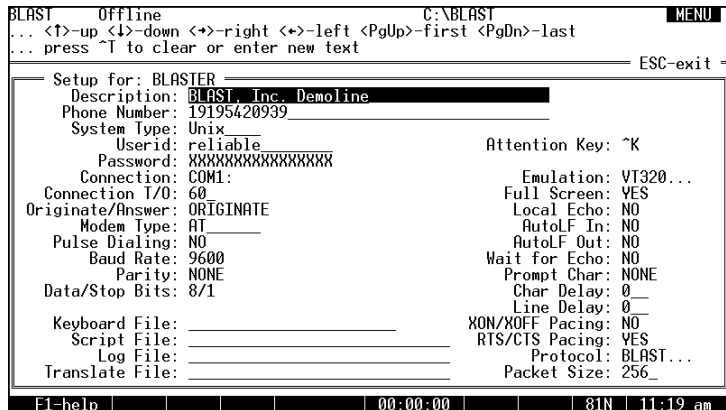
Creating a New Setup

To create a new setup, select the New option in the Offline menu by pressing **n**. BLAST will prompt with:

Enter new setup name: _____

Choose a name up to eight characters long. You may want to use the location of the remote site as the setup name. BLAST will automatically append the extension “.SU” to the filename. Type in the setup name and press **ENTER**. BLAST will enter Modify mode (see next section) and display in the Setup window (Figure 5-2) the values of default setup. After you modify these values and press **ESC**, BLAST will automatically save the new setup file, load its values into memory, and return to the Offline menu.

FIGURE 5-2



Modifying a Setup

To modify a setup file from the Offline menu, use the up and down arrow keys to highlight the setup file in the directory and then press **M**. This automatically loads the setup values into memory, overriding a previously loaded setup. You will see the Setup window (Figure 5-2, preceding page).

A field must be highlighted before you can modify its value. Use the arrow keys to move from field to field. The third line of the screen will display the type of action necessary to enter a value. Most fields are multiple choice. Use **SPACEBAR** to cycle forward and **BACKSPACE** to cycle backwards through the available options in these fields; then press **ENTER** to proceed to the next field. Some fields, such as Phone Number, require your input. To correct a mistake while entering data, use **BACKSPACE** to delete the mistake and then continue typing, or press **CTRL T** to clear the field and start over.

The Emulation and Protocol fields may require additional input. If the entry in the field is followed by three periods, e.g., “VT220...,” there is a subwindow of additional settings. Press **ENTER** to access the subwindow. After making the necessary changes to the subwindow, press the **ESC** key to return to the Setup window.

After finishing the modifications, press **ESC** and you will see the following prompt, “...save changes to the current setup?” If you choose “Yes,” **BLAST** will overwrite the old file with the new values. The original settings are lost when you overwrite a file; always use the **New** command (instead of **Modify**) when you do not wish to overwrite existing settings. There are two alternatives if you answer “No” to the save prompt:

1. If after choosing “No,” you select **Connect** from the Offline menu, you will be asked again if you wish to save the changes. If you answer “No,” **BLAST** will discard the changes you have made and connect you using the unmodified values in the setup.
2. If after choosing “No,” you select **Online** from the Offline menu, you will enter the Online menu. From the Online menu, if you select **Connect** or **Terminal**, **BLAST** will use the values that you have just modified in your setup, but it will not save them to disk. Before you can modify another setup or quit the program, **BLAST** will ask whether or not you want to save the current changes to your setup file.

Removing a Setup

To delete a setup, use the up and down arrow keys to highlight the setup, press **T** to tag the setup, and then press **R**. When you are prompted to delete the setup, select “Yes” to delete the setup or “No” to cancel the deletion.

Setup Fields

This section briefly discusses the function of each setup field of the Setup window and indicates default values in brackets and corresponding BLASTscript variables in italics (For more on BLASTscript variables, see Chapter 16). The Online Help for each field also contains detailed information. For quick reference, the page numbers for descriptions of major setup fields are listed below.

<u>FIELD</u>	<u>PAGE</u>	<u>FIELD</u>	<u>PAGE</u>
DESCRIPTION:	65	<i>DEC VT SUBWINDOW:</i>	74
PHONE NUMBER:	66	<i>WYSE SUBWINDOWS:</i>	78
SYSTEM TYPE:	66	<i>HP SUBWINDOW:</i>	82
USERID:	67	FULL SCREEN:	84
PASSWORD:	67	LOCAL ECHO:	84
CONNECTION:	67	AUTOLF IN:	84
CONNECTION T/O:	68	AUTOLF OUT:	85
ORIGINATE/ANSWER:	68	WAIT FOR ECHO:	85
MODEM TYPE:	69	PROMPT CHAR:	85
PULSE DIALING:	70	CHAR DELAY:	85
BAUD RATE:	70	LINE DELAY:	86
PARITY:	70	XON/XOFF PACING:	86
DATA/STOP BITS:	71	RTS/CTS PACING:	86
KEYBOARD FILE:	71	PROTOCOL:	87
SCRIPT FILE:	71	<i>BLAST SUBWINDOW:</i>	87
LOG FILE:	71	<i>KERMIT SUBWINDOW:</i>	92
TRANSLATE FILE:	71	<i>ZMODEM SUBWINDOW:</i>	95
ATTENTION KEY:	72	PACKET SIZE:	99
EMULATION:	72		
<i>ANSI SUBWINDOW:</i>	72		
<i>DG SUBWINDOW:</i>	73		

Description

user-defined

Provides a detailed description of the setup. This information is a free form comment; however, scripts can use the variable @SYSDDESC for any purpose. For example, the program can take in-

formation from the description line as input or write to it to save status information.

BLASTscript variable: @SYSDDESC

Phone Number user-defined

Stores the phone number of the remote computer. This field allows up to 40 characters. Although any alphanumeric character may be entered, be careful to avoid using characters that may be misinterpreted by the modem. Most modems allow commas to signify pauses within a dialing string. This string is passed unchanged to the modem. See your modem manual for details.

For a direct connection, leave the Phone Number field empty.

BLASTscript reserved variable: @PHONENO

System Type any valid system type

Identifies the multi-user computer type to which BLAST will connect. If you are connecting to a computer not listed here or to a single-user system, select NONE. (Mac and PC types are provided for consistency with BLAST scripts but are equivalent to NONE.) The Connect, Disconnect, Filetransfer, and Upload processes use this information to automate your logons and file transfers.

The available system types are modified periodically by BLAST, Inc. The the system types in the list below were the ones available at the time of publication. To obtain the most recent list, you may download the most recent system script from our FTP site at <ftp://blast.com/dist/scripts/>.

None – Single-user system such as IBM PC or Apple Macintosh
PC – IBM PC
Mac – Apple Macintosh
VMS – DEC VAX VMS
AOS – Data General AOS
BHost – BLAST Host
UNIX – UNIX
XENIX – Xenix
AIX – IBM RS/6000
A/UX – Apple UNIX
HP-UX – Hewlett-Packard UNIX
IRIX – Silicon Graphics UNIX
QNX – QNX 4.2
SCO – Santa Cruz Operation UNIX for PC

SunOS – Sun UNIX
Ultrix – DEC VAX Ultrix
CEO – Data General
MVS/TSO – IBM Mainframe
VM/CMS – IBM Mainframe
WBHOST – WinBLAST

To specify a user-defined system type, enter into this field the name of the .SCR file (which must reside in the current directory or in BLASTDIR). See Chapter 14 for more details on SYSTEMS.SCR and user-defined system scripts.

BLASTscript variable: @SYSTYPE

Userid user-defined

Holds the login ID used to log onto the remote system. With the value of this field, BLAST's CONNECT command uses the SYSTEMS.SCR library to answer logon queries automatically.

BLASTscript variable: @USERID

Password user-defined

Holds the password used to log onto the remote system. With the value of this field, BLAST's CONNECT command uses the SYSTEMS.SCR library to answer password queries automatically. To maintain security, this field is intentionally overwritten with Xs in the Setup window and encoded in the setup file on the disk.

As additional security, BLAST prompts you for this password if this field is left blank; therefore, the password need not be on the disk at all. For more information, see @PASSWORD on page 259.

BLASTscript variable: @PASSWORD

Connection any valid commport
or network device

Specifies the communications port or LAN driver and destination that BLAST will use. This field has two parts:

Connection Type – specifies either an asynchronous communications port (COMx: or BIOSx:) or a network driver assigned in BLAST.OPT. For serial connections, acceptable values are COM1: – COM8:, BIOS1:, BIOS2:, or any user-defined communications port. For LAN connections, acceptable values are network drivers

that were defined during the installation process or in a valid BLAST.OPT (for a list of drivers, see “BLAST Network Drivers” on page 28).

Connection Name – specifies the network destination name if Connection Type is a LAN driver (for example, a valid name would be `blaster(198.116.85.11)`, TCP/IP; see your Installation Guide for more information on destination names.)

NOTE: You may type a name at the bottom of the Connection Name subwindow. This destination name is used for the current session but is not added to your default list of Connection Names. The Connection Name is stored in @NETSERVICE.

To connect with a remote computer through an X.25 gateway, you must load the X.25 redirection software (such as Netware X.25 or Eicon X.25) on the network before you run BLAST. When you run BLAST, choose BIOS1: as your Connection.

The default value of this field is set during BLAST installation if LAN drivers and destination names are specified during installation.

BLASTscript variable: @COMMPORT (Connection Type)

BLASTscript variable: @NETSERVICE (Connection Name)

Connection T/O 0 – 999 [60]

For network connections, specifies the number of seconds that BLAST will wait for a network connection after entering the Online menu. This field has no effect on serial connections.

If the specified amount of time passes and a connection has not been made, BLAST will display an error message, set @STATUS to a non-zero value, and return to the Online menu.

If this field is set to 0, BLAST will not time out.

BLASTscript variable: @CONNTIMO

Originate/Answer [ORIGINATE] ANSWER

Specifies what BLAST will do during the automated connect and disconnect processes. To dial out and initiate a connection, set the field to ORIGINATE. To set BLAST to wait for a caller to connect, set the field to ANSWER.

NOTE: For node-to-node network connections, one system must be set to `ORIGINATE` and the other system set to `ANSWER`. A useful rule of thumb to maintain consistency is: If the Network Service name (in the Connection field) is your name, set your Answer/Originate field to `ANSWER`; if it is the name of the node you are trying to call, set the field to `ORIGINATE`.

BLASTscript variable: @ORGANS

Modem Type any valid modem type

Identifies the modem connected to your communications port. When you select the Online Connect or Disconnect menu command, or use the `CONNECT` or `DISCONNECT` BLASTscript command, BLAST uses the modem type named in this field to execute pre-defined programs from the `MODEMS.SCR` library. These routines perform various hardware-specific tasks, such as dialing the phone and disconnecting from the remote computer.

The available modem types are modified periodically by BLAST, Inc. The the modem types in the list below were the ones available at the time of publication. To obtain the most recent list, you may download the most recent `MODEMS.SCR` from our FTP site at <ftp://blast.com/dist/scripts/>.

- None – no modem specified
- Hardwire – direct connection
- Apex – Apex Data modems
- AT – AT command set
- AT&T – AT&T Paradyne modems
- Boca – Boca modems
- Cardinal – Cardinal modems
- Codex – Codex modems
- GVC – GVC modems
- Hayes – Hayes modems
- Intel – Intel modems
- MegaHz – Megahertz modems
- Microcom – Microcom modems
- Motorola – Motorola Universal Data Systems (UDS) modems
- Multitec – MultiTech modems
- Ositech – Ositech modems
- Practicl – Practical Peripherals modems
- Supra – Supra modems
- Telebit – Telebit modems
- UDSFasTk – Motorola UDS FasTalk
- UDSV3229 – Motorola UDS V3229
- USRobot – U.S. Robotics modems

USRV32 – U.S. Robotics Courier V.32, V.32bis, V.42, V.42bis
Zoom – Zoom modems
ZyXEL – ZyXEL modems

If your modem does not appear as a choice in the setup field, you may specify a user-defined modem type by entering into this field the name of the .SCR file. See Chapter 14 for more details on MODEMS.SCR and user-defined modem scripts. The default Modem Type is set during BLAST Installation.

BLASTscript variable: @MODEM

Pulse Dialing **YES [NO]**

Specifies whether to use pulse dialing. If this field is set to YES, pulse dialing is used; if it is set to NO, tone dialing is used.

BLASTscript variable: @PULSEDIAL

Baud Rate 300 600 1200 2400 4800
9600 19.2 38.4 57.6 115K

Indicates the speed of your PC's communications port, usually the same as your modem's top speed. Some older modems are incapable of negotiating baud rates with other modems. If you have trouble connecting with other systems, match your Baud Rate setting with the highest Baud Rate supported by the remote computer.

This is a multiple choice field. The default value is set during BLAST installation.

Sometimes it is advantageous to run at a lower than maximum baud rate. Slow PCs may drop characters at very high baud rates, causing garbled displays in Terminal mode and a high number of block re-transmissions during file transfers. Throughput may be better at a slower rate.

BLASTscript variable: @BAUDRATE

Parity **[NONE] EVEN ODD MARK SPACE**

Sets the parity of the communications port. This setting should match that of the remote computer.

BLASTscript variable: @PARITY

Data/Stop Bits

7/1 7/2 [8/1] 8/2

Sets the number of data bits (7 or 8) and number of stop bits (1 or 2) for the communications port while you are in Terminal mode.

BLASTscript variable: @D/S_BITS

Keyboard File

filename

Specifies a user-defined keyboard map for a particular keyboard or application. This field remains blank unless you create your own keyboard maps with BLASTKBD, the BLAST keyboard mapping utility (see “Keyboard Mapping Utility” on page 296).

BLASTscript variable: @KEYFILE

Script File

filename

Designates a BLAST script that is executed immediately when the user selects the Online menu. A script specified on the BLAST command line overrides a script specified in this field.

Use BLAST scripts to automate part or all of a BLAST session.

BLASTscript variable: @SCRFILE

Log File

filename

Names the log file that keeps a record of all session activity. When a file is transferred, a menu selection made, or a BLASTscript statement executed, the log file records the activity and the time that it occurred. Extended Logging offers detailed information about file transfers. For more information on Extended Logging, see the description of the @XLOG reserved variable on page 278.

If the filename that you enter already exists, BLAST appends the new session activity information to the existing file; otherwise the file is created. Log files do not need any particular extension and can be any combination of the normally accepted filename characters. You may specify a full path as part of the log filename.

BLASTscript variable: @LOGFILE

Translate File

filename

Designates a control file to filter incoming or outgoing characters in Terminal mode and during text upload and capture. The Translate File is an ASCII text file that can be edited by a text processor or the

BLAST editor. See “Translate File Format” on page 284 for more information.

BLASTscript variable: @XLTFILE

Attention Key any Control key [**^K**]

Defines the key combination interpreted as the Attention Key. This field accepts a single keystroke, which is used in combination with the CTRL key. Throughout this manual, the Attention Key is referred to as *ATTN*.

If the value of the Attention Key is changed, the replacement value must be carefully chosen. Certain sequences are special codes used for text formatting and management. For instance, the ENTER key sends a ^M as the code for a carriage return. If ^M is used for the attention key, the ENTER key will also function as *ATTN*, usually with undesired results: if you press the ENTER key twice in Terminal mode, you will escape to the Online menu.

We recommend that you do not change this setting.

BLASTscript variable: @ATTKEY

Emulation any valid terminal emulator [**VT320**]

Specifies the terminal emulator for the communications session. This is a multiple-choice field.

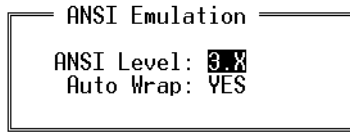
The possible values for this multiple choice field are VT320, VT220, VT100, VT52, ANSI, D461, D411, D410, D200, TV920, D80, ADM3A, WYSE50, WYSE60, HP2392, IBM3101, and TTY. BLAST also supports pass-through terminal emulation, in which the characters received on the communications port are displayed without change. For complete transparency, set this field to TTY and set the setup Translate File field to PASSTHRU.TBL.

BLASTscript variable: @EMULATE

ANSI Emulation Subwindow

Selecting the ANSI emulator and pressing ENTER displays the ANSI Emulation subwindow shown in Figure 5-3 on the next page:

FIGURE 5-3



ANSI Level

2.x **[3.x]**

Specifies the correct level of ANSI for your system. Some applications require ANSI Level 2 . x.

BLASTscript variable: @ANSILEVEL

Auto Wrap

[YES] NO

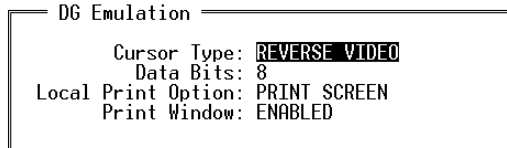
Specifies automatic wrapping of lines longer than 80 characters.

BLASTscript variable: @ANSIAUTOWRAP

DG Emulation Subwindow

Selecting any of the Data General (DG) emulators and pressing ENTER displays the DG Emulation subwindow shown below in Figure 5-4:

FIGURE 5-4



Cursor Type

NONE UNDERLINE
[REVERSE VIDEO]

Determines whether the cursor is displayed as a reverse-video block, as an underline character, or not displayed.

BLASTscript variable: @DGCURSTYPE

Data Bits

7 **[8]**

Sets data bits.

BLASTscript variable: @DGDATABITS

Local Print Option

[PRINT SCREEN]
PRINT WINDOW
PRINT PASSTHROUGH ON
SIMULPRINT ON PRINT FORM

Sets local print key action.

BLASTscript variable: @DGPRTMODE

Print Window

[ENABLED] DISABLED

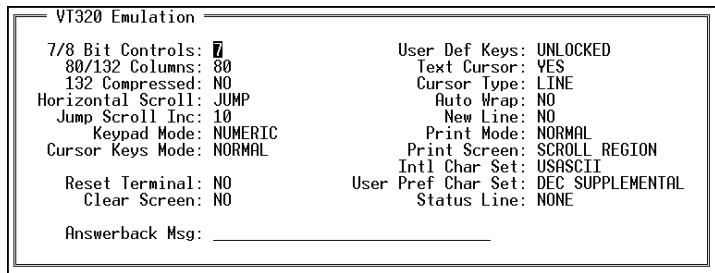
Specifies whether the local print key and keyboard shortcuts for DG local print commands are enabled.

BLASTscript variable: @DGPRTWIND

DEC VT Emulation Subwindows

Selecting any of the VT emulators and pressing ENTER will display a VT subwindow of extended configuration options. The VT320 subwindow is shown below in Figure 5-5. The VT220 and VT52/100 subwindows are subsets of the VT320 subwindow.

FIGURE 5-5



7/8 Bit Controls

[7] 8

Specifies whether “CI” control characters are represented in the 8-bit environment or as 7-bit escape sequences.

BLASTscript variable: @VT8BIT

80/132 Columns

[80] 132

Specifies 80-column or 132-column display for text.

BLASTscript variable: @VTDISP132

132 Compressed

YES [NO]

Specifies compressed mode for video when the 80/132 Columns setup field is set to 132 or the host sends a sequence to the emulator to use 132 columns. To use this feature, your PC must be equipped with an EGA or VGA adapter card and compatible monitor.

BLASTscript variable: @VTCOMPRESSED

Horizontal Scroll

[JUMP] SMOOTH NONE

Specifies how to scroll data on an 80-column display when the emulator is in 132-column mode. SMOOTH moves the view of the display only as necessary to display the cursor position. JUMP adjusts the view based on the setting of the setup field Jump Scroll Inc. When NONE is selected, the display does not scroll and the cursor may disappear from view. The value of this field is ignored when 132 Compressed setup field is set to YES.

BLASTscript variable: @VTHSCROLL

Jump Scroll Inc

1 – 53 [10]

Specifies the number of columns to scroll left or right when the cursor reaches the edge of the screen and the Horizontal Scroll setup field is set to JUMP.

BLASTscript variable: @VTHSCROLLN

Keypad Mode

[NUMERIC] APPLICATION

Specifies whether the numeric keypad keys will send numbers (NUMERIC) or programming functions defined by the application (APPLICATION).

BLASTscript variable: @VTKEYPAD

Cursor Keys Mode

[NORMAL] APPLICATION

Specifies whether the cursor keys will control cursor movement (NORMAL) or send application control functions (APPLICATION).

BLASTscript variable: @VTCURSOR

Reset Terminal

YES [NO]

Specifies resetting many of the VT320 operating features, such as scrolling regions and character attributes, to their factory default val-

ues upon entering Terminal mode. YES resets these values; the value of this variable is then automatically reset to NO.

BLASTscript variable: @VTRESET

Clear Screen YES **[NO]**

Specifies clearing of the terminal's video display the next time you enter Terminal mode. YES clears the terminal's video display; the value of this variable is then automatically reset to NO.

BLASTscript variable: @VTCLRSCRN

Answerback Msg up to 30 characters

Contains a message to be sent to the remote computer upon receiving an inquiry (^E). The message can be up to 30 characters long.

BLASTscript variable: @VTANSBACK

User Def Keys **[UNLOCKED]** LOCKED

Specifies whether the host system can change user-defined key (UDK) definitions.

BLASTscript variable: @VTUSERKEYS

Text Cursor **[YES]** NO

Specifies whether to display the text cursor.

BLASTscript variable: @VTTEXTCURS

Cursor Type BLOCK **[LINE]**

Specifies whether the cursor is displayed as a reverse-video block or as an underline character.

BLASTscript variable: @VTCURSTYPE

Auto Wrap YES **[NO]**

Specifies whether text typed at the right margin will automatically wrap to the next line.

BLASTscript variable: @VTAUTOWRAP

New Line

YES [NO]

Specifies whether the ENTER key will move the cursor to a new line. Possible choices are NO (the ENTER key sends only a carriage return) and YES (both a carriage return and line feed are sent).

BLASTscript variable: @VTNEWLINE

Print Mode [NORMAL] AUTO CONTROLLER

Specifies when information is sent to the printer. In AUTO print mode, each line of received text is displayed and printed; in CONTROLLER mode, all received data is sent directly to the printer without displaying it on the screen; and in NORMAL mode, the user initiates printing from the keyboard.

BLASTscript variable: @VTPRINT

Print Screen [SCROLL REGION] FULL PAGE

Specifies how much of the screen to print when you press the PRINT SCREEN key. Choices are FULL PAGE (entire page) and SCROLL REGION (only the currently defined VT scrolling region).

BLASTscript variable: @VTPRINTPAGE

Intl Char Set

**[USASCII] UK FRENCH
GERMAN ITALIAN
SPANISH DANISH**

Specifies whether 7- or 8-bit data is used for international support. The default value, USASCII, allows 8-bit data. The high-order values are used to represent international characters. If any other character set is selected, specific international characters replace characters within the ASCII set.

BLASTscript variable: @VTINTL

User Pref Char Set

**[DEC SUPPLEMENTAL]
ISO LATIN-1**

Specifies either DEC SUPPLEMENTAL or ISO LATIN-1 as the user preferred character set.

BLASTscript variable: @VTUSERCHAR

Status Line

[NONE] INDICATOR
HOST WRITABLE

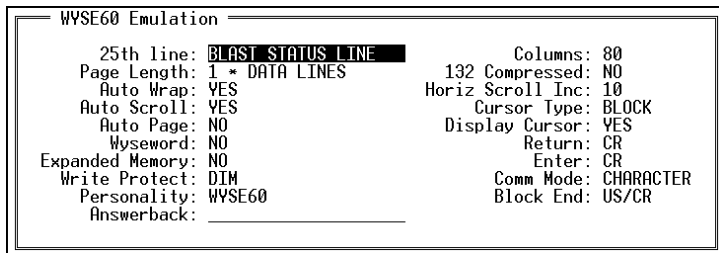
Specifies the status line to be displayed at the bottom of the screen. The INDICATOR status line displays cursor position, printer status, and modem status information; the HOST WRITABLE status line displays messages from the Host computer. Selecting NONE specifies the BLAST status line.

BLASTscript variable: @VTSTATUSLN

WYSE Emulation Subwindows

Selecting the WYSE60, WYSE50, TV920, D80, or ADM3A emulator and pressing ENTER will display one of two subwindows of extended configuration options: the WYSE60 Emulation subwindow (Figure 5-6 below) or the WYSE50/TV/D80/ADM3A subwindow. These subwindows are identical except for the Personality option available with WYSE60 emulation.

FIGURE 5-6



25th Line

[BLAST STATUS LINE]
25th DATA LINE
STANDARD STATUS LINE
EXTENDED STATUS LINE

Specifies how the 25th line will be used. 25th DATA LINE specifies no status line—the 25th line will be used for data. BLAST STATUS LINE specifies the BLAST status line. STANDARD STATUS LINE specifies the standard Wyse status line. EXTENDED STATUS LINE specifies the extended Wyse status line. Both the standard and extended status lines display keyboard lock information, current page number, Wyseword mode, communication mode, flow control information, printer status, and a host-definable computer message. The standard status line also displays the current

time and cursor position, and the extended status line also displays insert mode and field protection status.

BLASTscript variable: @WY25LINE

Page Length **[1 * DATA LINES]**
2 * DATA LINES 4 * DATA LINES

Specifies page length in number of screens of data. 2* DATA LINES sets the page length to 48 lines or 50 lines depending on the value specified in the 25th line setup field.

BLASTscript variable: @WYPAGELEN

Auto Wrap **[YES] NO**

Specifies automatic line wrapping.

BLASTscript variable: @WYAUTOWRAP

Auto Scroll **[YES] NO**

Specifies scrolling of the terminal display when the cursor reaches the bottom of a page. If Auto Scroll is set to OFF, the cursor is placed at the home position instead of scrolling. The Auto Scroll value is ignored if Auto Page is set to ON.

BLASTscript variable: @WYAUTOSCROLL

Auto Page **YES [NO]**

Specifies whether the cursor can move off the current page. If YES is selected, the cursor can move above the first line to the previous page or below the last line to the next page.

BLASTscript variable: @WYAUTOPAGE

Wyseword **YES [NO]**

Specifies whether keys send Wordstar™ functions (YES) or the standard key codes (NO). (The only keys that are affected are the Wyse keys that can be mapped with the BLASTKBD utility; see “Keyboard Mapping Utility” on page 296).

BLASTscript variable: @WYSEWORD

Expanded Memory YES **[NO]**

Specifies whether “expanded” memory is used. Note that this setting is not related to DOS expanded memory. Normally, the terminal emulator uses two pages of video display memory. If the maximum of four pages is required, expanded memory must be set to YES. Note, however, that more run-time memory will be required by BLAST, possibly adversely affecting throughput during file transfers.

BLASTscript variable: @WYEXPNDMEM

Write Protect **[DIM]** REVERSE NORMAL

Specifies the attributes used to display protected fields.

BLASTscript variable: @WYWRITEPROT

Personality **[WYSE60]** (WYSE60 Only) PC Term

Specifies WYSE60 or PC Term personality.

BLASTscript variable: @WYPERSONALITY

Answerback up to 20 characters

Contains a message to be sent to the remote computer upon receiving an inquiry (^E). The message can be up to 20 characters long.

BLASTscript variable: @WYANSBACK

Columns **[80]** 132

Specifies 80 or 132 columns per row.

BLASTscript variable: @WYDISP80

132 Compressed YES **[NO]**

Specifies whether compressed mode is used when the Columns set-up field is set to 132 or the host sends a sequence to the emulator to use 132 columns. To use this feature, your PC must be equipped with an EGA or VGA adapter card with 132 column support and a compatible monitor.

BLASTscript variable: @WYCOMPRESSED

Specifies the number of columns to scroll when the cursor reaches the edge of the screen and the Columns setup field is set to 132 and the 132 Compressed setup field is set to NO. Note that a value of 1 specifies smooth scrolling. Any other value specifies jump scrolling.

BLASTscript variable: @WYSCROLLING

Cursor Type**[BLOCK] LINE**

Specifies whether the cursor is displayed as a reverse-video block or as an underline character.

BLASTscript variable: @WYCURSTYPE

Display Cursor**[YES] NO**

Specifies whether the cursor is visible.

BLASTscript variable: @WYDSPCURSOR

Return**[CR] CRLF TAB**

Specifies the character to send when the RETURN key is pressed.

BLASTscript variable: @WYRETURN

Enter**[CR] CRLF TAB**

Specifies the character to send when the keypad ENTER key is pressed.

BLASTscript variable: @WYENTER

Comm Mode**[CHARACTER] BLOCK**

Specifies whether data is sent after each keystroke (CHARACTER mode) or grouped in blocks (BLOCK mode).

BLASTscript variable: @WYCOMMmode

Block End**[US/CR] CRLF/ETX**

Specifies what characters mark the end-of-line and end-of-block when the terminal is in block mode and sends a block of data. If

US/CR is selected, a US character (\037) is sent at the end of each line and a CR character (\015) is sent to mark the end of the block.

BLASTscript variable: @WYBLOCKEND

HP Emulation Subwindow

Selecting the HP2392 emulator and pressing ENTER will display the HP Emulation subwindow shown below in Figure 5-7:

FIGURE 5-7

```
HP Emulation
Terminal Id: ██████      Start Col: 0
InhHndShk(G): NO       Line/Page(D): LINE
Inh DC2(H): NO         FldSeparator: ^~
InhEolWrp(C): NO      BlkTerminator: ^~
Destructive BS: NO
```

Terminal Id [2392] any valid terminal ID

Specifies what terminal identification will be sent to the remote computer when a Terminal Id request (ESC *s^) is made.

BLASTscript variable: @HPTERMID

InhHndShk(G) [NO] YES

Specifies inhibition of D1 handshaking when data is transferred from the emulator to the remote computer.

InhHndShk (G) is used with Inh DC2(H) setup field to determine what type of handshaking is used. If InhHndShk(G) is set to YES, and Inh DC2(H) is set to NO, D1/D2/D1 handshaking is used. If both InhHndShk(G) and Inh DC2(H) are set to YES, all handshaking is inhibited.

BLASTscript variable: @HPINHHNDSHK

Inh DC2(H) [NO] YES

Specifies inhibition of D1/D2/D1 handshaking when data is transferred from the emulator to the remote computer.

Inh DC2(H) is used with InhHndShk(G) setup field to determine what type of handshaking is used. If Inh DC2(H) is set to YES, and InhHndShk(G) is set to NO, D1 handshaking is used. If both

InhHndShk(G) and Inh DC2(H) are set to YES, all handshaking is inhibited.

BLASTscript variable: @HPINHDC2

InhEolWrp(C) **[NO]** YES

Specifies inhibition of automatic text wrapping. If this field is set to NO, text automatically wraps; if it is set to YES, when the cursor reaches the right margin, it remains there (with succeeding characters overwriting the existing character) until a carriage return or other cursor movement is performed.

BLASTscript variable: @HPINHWRAP

Destructive BS **[NO]** YES

Specifies that BACKSPACE erase the character under the cursor after moving the cursor one character to the left.

BLASTscript variable: @HPDESTRBS

Start Col **[0]** YES

Specifies the position of the left margin if no logical start-of-text pointer has been generated.

BLASTscript variable: @HPSTARTCOL

Line/Page(D) **[LINE]** PAGE

Specifies whether a line or a page at a time is transmitted in block mode.

BLASTscript variable: @HPLINEPAGE

FldSeparator any Control character **[^]**

Specifies the field separator character. When you press ENTER while in block page mode containing a formatted display, a field separator character is automatically transmitted at the end of each protected field (except the final one).

BLASTscript variable: @HPFLDSEP

BlkTerminator any Control character [^^]

Specifies the block terminator character. A block terminator character is transmitted to the remote computer at the end of a transfer operation.

BLASTscript variable: @HPBLKTERM

END OF EMULATION SUBWINDOW DESCRIPTIONS

Full Screen [YES] NO

Indicates whether the top four lines of the menu display will be suppressed in Terminal mode. The default value, YES, suppresses the menu and allows the top 24 lines of the terminal screen to be used for data.

BLASTscript variable: @FULLSCR

Local Echo YES [NO]

Specifies whether BLAST will echo typed characters to the screen in Terminal mode. If this field is set to YES, BLAST will display typed characters before sending them out the communication port; if the field is set to NO, BLAST will display characters only if the remote computer sends them back.

If this field is set to YES and double characters are displayed on the screen, change the setting to NO.

BLASTscript variable: @LOCECHO

AutoLF In YES [NO]

Controls the Terminal mode actions when receiving carriage returns. Some remote systems do not automatically supply line feeds, causing multiple lines of text written on top of each other on your monitor. To read incoming text correctly from this computer type, set AutoLF In to YES. The setting for AutoLF In has no effect on text received in Capture mode.

BLASTscript variable: @AUTOLFIN

AutoLF Out

YES [NO]

Controls Terminal mode actions when sending carriage returns. A setting of YES causes BLAST to append a line feed to each carriage return sent out from the communications port. Line feeds are often stripped from the data stream to increase throughput. If the remote system requires a line feed after the carriage return, set AutoLF Out to YES.

BLASTscript variable: @AUTOLFOUT

Wait for Echo

YES [NO]

During text uploads, specifies that BLAST wait for the echo of the previously sent character before sending another character; this setting has no effect on file transfers.

Wait for Echo “paces” text uploads to slow BLAST down when the remote computer operates more slowly than the local system. It is also useful when sending one line commands to modems that cannot take bursts of high speed data while in Command mode.

BLASTscript variable: @WT4ECHO

Prompt Char

[NONE] any ASCII character

Specifies the character that BLAST will use to determine when to resume sending text. After sending a line of text and a carriage return, BLAST pauses until the remote system sends the prompt character. Prompting is an effective form of flow control while uploading text.

Any single character, including a control character, is a valid entry. To enter a control character, prefix the character with a caret (^). NONE disables prompting.

BLASTscript variable: @PROMPTCH

Char Delay

[0] – 999

Specifies the time period (in hundredths of a second) that BLAST pauses between sending each character to the remote computer. This pause slows down strings sent by BLAST scripts and text that is uploaded.

Character delay is a form of flow control. Use Char Delay when the remote computer is unable to keep pace with BLAST and no other form of flow control is available or to slow down the interaction with a modem or other simple hardware device that does not support oth-

er forms of flow control. The default value, 0, specifies no delay. Char Delay delay applies only to text uploads; it has no effect on file transfers.

If you are performing a text upload over a connection that is not 100% error free, it may be necessary to set Char Delay to a value other than 0 to upload text reliably.

BLASTscript variable: @CHARDLY

Line Delay **[0] – 999**

Specifies the length of time (in tenths of a second) to pause after sending a line of data. Line Delay provides a form of flow control while uploading text to the remote computer. Some remote systems may be unable to keep pace with BLAST; setting this field to a non-zero value can prevent overloading the remote computer. If Line Delay is set to 0, no delay will occur. Line Delay applies only to text uploads; it has no effect on file transfers.

If you are performing a text upload over a connection that is not 100% error free, it may be necessary to set Line Delay to a value other than 0 to upload text reliably.

BLASTscript variable: @LINEDLY

XON/XOFF Pacing **YES [NO]**

Specifies whether BLAST will use software flow control during text uploading, Terminal mode operation, and file transfer. When one computer needs to stop the flow of incoming data, it transmits XOFF (CTRL S) to the other computer. When the computer is again ready to receive data, it transmits XON (CTRL Q).

During BLAST protocol transfer, BLAST will wait a maximum of 30 seconds for an XON from the remote. If the XON is not sent, BLAST will resume transfer. For a detailed discussion of flow control, see page 35.

BLASTscript variable: @XONXOFF

RTS/CTS Pacing **[YES] NO**

Enables hardware flow control. RTS/CTS pacing uses the RS-232 signals Request-to-Send and Clear-to-Send for optimized throughput over error-correcting modems. Set this field to NO unless error-

correcting modems are on both ends of the connection. For a detailed discussion of flow control, see page 35.

BLASTscript variable: @RTSCTS

Protocol

[BLAST] KERMIT
XMODEM XMODEM1K
YMODEM YMODEMG
ZMODEM FTP

Specifies the protocol to be used for file transfers. BLAST protocol generally runs faster and offers more features than other protocols.

BLASTscript variable: @PROTOCOL

BLAST Protocol Subwindow

Selecting BLAST and pressing ENTER displays the BLAST Protocol subwindow shown in Figure 5-8 below:

FIGURE 5-8

BLAST Protocol	
Logon T/O: <u>120</u>	Retransmit timer: 4_
Inactivity T/O: 120	ACK Request Frequency: 4_
Transfer Password: _____	Number of Disconnect Blocks: 3
7 Bit Channel: NO	Launch String: Vr _____
Window Size: 16	Enable /FWD and /STR: NO
DCD Loss Response: IGNORE	Enable /OVW and Remote Cnds: YES
Use "A" Protocol: NO	Send Compression Level: 4
Filtering: OFF	Receive Compression Level: 4

Logon T/O

0 – 999 [120]

Specifies the number of seconds that BLAST will attempt to establish a filetransfer session with the remote computer. Logon Timeout affects BLAST protocol transfers and remote control sessions.

Timeouts can happen if:

- ◇ There is excessive noise on the line.
- ◇ There are parity or data/stop bit mismatches.
- ◇ BLAST is terminated unexpectedly on the remote computer.
- ◇ The connection is lost.

If Logon T/O is set to 0, no timeout will occur and BLAST will attempt to establish a filetransfer session with the remote computer indefinitely.

BLASTscript variable: @LOGTIMO

Inactivity T/O **0 – 999 [120]**

Defines the time interval (in seconds) that BLAST will stay connected after the last valid data packet has been received from the remote computer. Timeouts happen if:

- ◇ The connection is lost.
- ◇ There is excessive noise on the line.
- ◇ The remote computer goes down.
- ◇ Flow control has not been released.

If Inactivity T/O is set to 0, BLAST never times out.

NOTE: In previous versions of BLAST, this field was named “Connect Timeout” and was associated with the BLASTscript reserved variable @CONTIMO.

BLASTscript variable: @INACTIMO

Transfer Password **user-defined**

Stores a case-sensitive password (up to eight characters) that restricts a remote user’s access. Requests to get files from a password-protected computer and to do file maintenance functions are not honored unless the password is received first. Without the password, the remote machine is limited to sending and receiving messages.

To send the Transfer Password, the remote user should select the Send menu command from the Filetransfer menu; then, at the local filename prompt, type the following:

```
!password=your_password
```

where *your_password* is the transfer password. The remote filename field and transfer options should be left blank. In a BLAST script, the SEND statement should be followed by a line with the password and then two blank lines.

NOTE: The Transfer Password is intended to validate remote users logging onto your system. If a local operator uses a setup with a Transfer Password entered, he or she will not be able to receive files without the remote computer sending the password.

BLASTscript variable: @TRPASSWD

7-Bit Channel

YES [NO]

Defines the logical width of the data path to be used. YES specifies a 7-bit data encoding scheme; NO specifies an 8-bit encoding scheme.

Some networks, minicomputers, and asynchronous devices only support 7-bit path widths. BLAST protocol operates more efficiently using 8-bit encoding; however, the data path width has nothing to do with the type of data that can be transferred. BLAST protocol may transfer 8-bit binary or 7-bit ASCII over either 7- or 8-bit data paths.

BLASTscript variable: @7BITCHN

Window Size

1 – [16]

Specifies the number of packets that can be sent to the remote without BLAST's waiting for an acknowledgement from the remote. As packets are acknowledged, the starting point of the window adjusts, or "slides." For example, if the window size is 12 and the first 6 of 8 packets sent have been acknowledged, the starting point of the window moves by 6, and 10 additional packets can be sent before BLAST must stop and wait for an acknowledgement.

BLASTscript variable: @WDWSIZ

DCD Loss Response

ABORT [IGNORE]

Specifies the action BLAST will take after DCD loss during a file-transfer session:

ABORT – Sets @EFERROR on carrier loss and exits Filetransfer mode.

IGNORE – Ignores carrier loss. Filetransfer mode continues until the Inactivity T/O takes effect.

BLASTscript variable: @DCDLOSS

Use “A” Protocol

YES [NO]

Specifies whether the BLAST “A” Protocol will be used. YES specifies communication with older BLAST products.

BLASTscript variable: @APROTO

Filtering

ON [OFF]

Specifies filtering out VT sequences sent from a remote computer or protocol converter. This filtering prevents BLAST protocol from labeling these sequences as bad blocks received.

BLASTscript variable: @FILTER

Retransmit Timer

0 – 9999 [4]

Sets the maximum number of seconds BLAST will pause before re-sending a packet. For example, if Window Size is set to 5 and Retransmit Timer is set to 30, BLAST will attempt to resend the fifth packet every thirty seconds if it receives no acknowledgement.

NOTE: This setting should be less than the that for Inactivity T/O.

BLASTscript variable: @RETRAN

ACK Request Frequency 1 – window size [4]

Specifies the frequency at which an acknowledgement from the receiving system is requested. The frequency is measured in number of packets sent. For example, if the ACK Request Frequency is 4, a request for an acknowledgement is sent to the receiving computer every four packets. Set this field higher for better performance with error-correcting modems. See also Window Size setup field on page 89.

BLASTscript variable: @ACKFREQ

Number of Disconnect Blocks

0 – 9 [3]

Set the number of additional disconnect blocks (after the first disconnect block) that BLAST sends when exiting Filetransfer mode. The default value, 3, specifies four total disconnect blocks.

BLASTscript variable: @NUMDISC

Launch String any ASCII string **[nr]**

Specifies a string to be appended to BLAST protocol packets. Appending this string will help communications to a mainframe through protocol converters. You may send any string of ASCII characters, including the same control characters used in string constants. Nonprintable characters can be represented with a backslash followed by a three-digit octal number (for example, a line feed may be represented as a \012). The string should not be enclosed in quotes.

BLASTscript variable: @LAUNCHST

Enable /FWD and /STR YES **[NO]**

Enables the /FWD and /STR file transfer switches. Note that disabling these switches affects only *local* files. For example, you will still be able to get a file with the /FWD switch because the successfully transferred file will be deleted from the *remote* system.

BLASTscript variable: @ENABLEFS

Enable /OVW and Remote Cmds **[YES] NO**

Enables the /OVW file transfer switch and system commands received during BLAST Protocol Filetransfer mode. Disabling /OVW affects only *local* files. For example, you will still be able to send a file with the /OVW switch because the file will be overwritten on the *remote* system. The List, Type, and More commands remain active when this field is set to NO; only potentially destructive commands are disabled.

BLASTscript variable: @ENABLERCMD

Send Compression Level 0 – 6 **[4]**

Specifies the maximum compression level that can be used while sending files to the remote computer. Level 0 specifies no compression; level 6 specifies the highest compression level (see “Compression Levels” on page 121).

BLASTscript variable: @SCOMP_LEV

Receive Compression Level 0 – 6 **[4]**

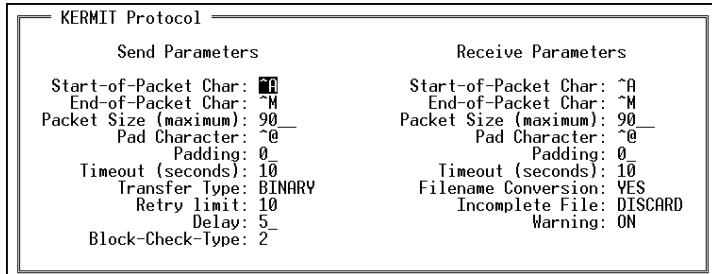
Specifies the maximum compression level that can be used while receiving files from the remote computer. Level 0 specifies no com-

pression; level 6 specifies the highest compression level (see “Compression Levels” on page 121).

BLASTscript variable: @RCOMP_LEV

Kermit Protocol Subwindow

Selecting KERMIT and pressing ENTER displays the Kermit Protocol subwindow shown in Figure 5-9 below:



The screenshot shows a window titled "KERMIT Protocol" with two columns of settings. The left column is labeled "Send Parameters" and the right column is labeled "Receive Parameters".

Send Parameters	Receive Parameters
Start-of-Packet Char: ^A	Start-of-Packet Char: ^A
End-of-Packet Char: ^M	End-of-Packet Char: ^M
Packet Size (maximum): 90	Packet Size (maximum): 90
Pad Character: ^@	Pad Character: ^@
Padding: 0	Padding: 0
Timeout (seconds): 10	Timeout (seconds): 10
Transfer Type: BINARY	Filename Conversion: YES
Retry limit: 10	Incomplete File: DISCARD
Delay: 5	Warning: ON
Block-Check-Type: 2	

FIGURE 5-9

Start-of-Packet Char [^A] – ^Z

For sending files with Kermit, specifies a control character to precede each packet sent from your PC. The same control character must also be used by the remote Kermit.

BLASTscript variable: @KSSOPKT

For receiving files with Kermit, specifies a control character to precede each packet received by your PC. The same control character must also be used by the remote Kermit.

BLASTscript variable: @KRSOPKT

End-of-Packet Char ^A – ^Z [^M]

For sending files with Kermit, specifies a control character to terminate each packet sent from your PC. The same control character must also be used by the remote Kermit.

BLASTscript variable: @KSEOPKT

For receiving files with Kermit, specifies a control character to terminate each packet received by your PC. The same control character must also be used by the remote Kermit.

BLASTscript variable: @KREOPKT

Packet Size

10 – 2000 [90]

For sending files with Kermit, specifies the packet size that your PC will use when it transmits a file. Note that the remote Kermit server's Receive Packet Size should also be set to this value. The larger the packet, the more efficient the transfer; however, larger packets will pose problems on a noisy connection. Set larger packet sizes when there is little line noise, you are communicating with an IBM mainframe, or you are using V.29 "ping pong" modems.

BLASTscript variable: @KSPKTLEN

For receiving files with Kermit, specifies the packet size that your PC will use when it receives a file. Note that the remote Kermit server's Send Packet Size should also be set to this value. The larger the packet, the more efficient the transfer; however, larger packets will pose problems on a noisy connection. Set larger packet sizes when there is little line noise, you are communicating with an IBM mainframe, or you are using V.29 "ping pong" modems.

BLASTscript variable: @KRPKTLEN

Pad Character

A – ^Z [^@]

For sending files with Kermit, specifies an alternate character to pad each packet transmitted by your PC.

BLASTscript variable: @KSPADCH

For receiving files with Kermit, specifies an alternate character to pad each packet received by your PC.

BLASTscript variable: @KRPADCH

Padding

0 – 99 [0]

For sending files with Kermit, specifies the number of padding characters to send per packet. Padding can induce delays during a Kermit file transfer, allowing slower machines or older versions of Kermit more time to process the data you send.

BLASTscript variable: @KSPADDNG

For receiving files with Kermit, specifies the number of padding characters to request per packet. Padding can induce delays during a Kermit file transfer, allowing slower machines or older versions of Kermit more time to process the data you receive.

BLASTscript variable: @KRPADDNG

Timeout **0 – 99 [10]**

For sending files with Kermit, specifies the number of seconds that your PC will wait after transmitting a packet before attempting to re-send it.

BLASTscript variable: @KSTIMEOUT

For receiving files with Kermit, specifies the number of seconds that your PC will wait to receive a packet before requesting it be resent.

BLASTscript variable: @KRTIMEOUT

Transfer Type **[BINARY]**

BLAST's implementation of Kermit supports only binary files; therefore, the value for this field is always BINARY and cannot be changed. Text files may be transferred with Kermit, but no special translation will be performed on them. To translate text file formats between computers with different platforms, use BLAST protocol.

BLASTscript variable: @KFILETYP

Retry Limit **1 – 99 [10]**

Specifies the number of times Kermit will attempt to send a packet before aborting. Set this field higher if the connection is intermittently noisy.

BLASTscript variable: @KRETRY

Delay **[5]**

Specifies the number of seconds of delay between the recognition of a Send command and the actual beginning of the transmission. The default value of 5 cannot be changed in BLAST.

BLASTscript variable: @KDELAYOS

Block-Check-Type

1 – 3 [2]

Specifies the level of error detection. Kermit offers three levels of error detection, with 3 being the most secure. Level 2 or 3 decrease the chance of a bad packet being accepted by the receiving computer but slows file transfer appreciably. Specify a lower level when using error-correcting modems or when transferring files at 9600 baud and above.

BLASTscript variable: @KBCHECK

Filename Conversion

[YES] NO

Specifies whether to convert a filename from local format to common Kermit format. For example, lower case is changed to all uppercase; and “~”, “#”, and all periods after the initial one are converted to “X”s.

BLASTscript variable: @KFNAMCONV

Incomplete File

[DISCARD] KEEP

Specifies whether to KEEP or DISCARD files incompletely received, such as a file being transferred when you abort a Get command. Setting this field to DISCARD insures that any file received is complete.

BLASTscript variable: @KSAVEINC

Warning

[ON] OFF

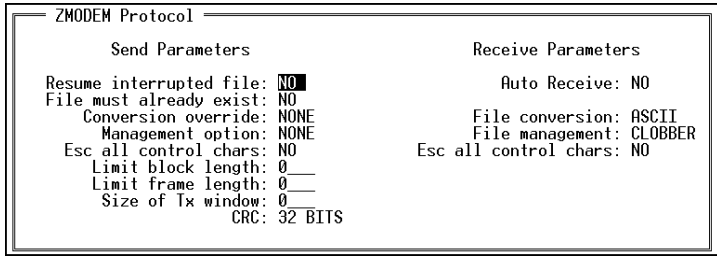
For Kermit transfers, specifies whether Kermit will automatically rename a received file if another file with the same name already exists in the current directory. If the field is set to ON, Kermit will rename the file by adding a number (0001, 0002, etc.) to the original filename; if the field set to OFF, Kermit overwrites the file.

BLASTscript variable: @KWARNING

Zmodem Protocol Subwindow

Selecting ZMODEM and pressing ENTER displays the Zmodem subwindow shown in Figure 5-10 on the next page:

FIGURE 5-10



Resume Interrupted File YES **[NO]**

Continues an aborted binary file transfer from the point of interruption. The destination file must already exist and be smaller than the source file.

BLASTscript variable: @ZMRESUME

File Must Already Exist YES **[NO]**

Transfers the file only if it already exists on the destination system.

BLASTscript variable: @ZMEXIST

Conversion Override **[NONE]** ASCII BINARY

Allows the sender to specify to the receiver whether the data should be treated as BINARY or ASCII data, overriding the File Conversion setting of the receiving system. If NONE is selected, the data is handled according to the receiver's file conversion parameter.

BLASTscript variable: @ZMCONVS

Management Option **[NONE]** PROTECT
CLOBBER NEWER
NEWER/LONGER DIFFERENT APPEND

Specifies a file management option for files sent. Possible values are:

NONE – The file is transferred if it does not already exist on the receiving system.

PROTECT – The file is transferred only if it does not already exist on the receiving system even if the receiving system has specified CLOBBER.

CLOBBER – The file is transferred whether or not it already exists on the receiving system unless the receiving system has specified PROTECT.

NEWER – The file is transferred if it does not already exist on the receiving system or if the source file is newer (by date).

NEWER/LONGER – The file is transferred if it does not already exist on the receiving system or if the source file is newer (by date) or longer (in bytes).

DIFFERENT – The file is transferred if it does not already exist on the receiving system or if the files have different lengths or dates.

APPEND – The file is appended to a file of the same name on the receiving system based on the value of the receiving system’s Receive File Conversion setting.

BLASTscript variable: @ZMMANAGS

Esc All Control Chars **YES [NO]**

For sending files with Zmodem, specifies that all control characters sent will be link-escape encoded for transparency. By default, only the characters represented by hexadecimal 10, 11, 13, 90, 91, and 93, and the sequence “@-CR” are link-escape encoded.

BLASTscript variable: @ZMCTLESCS

For receiving files with Zmodem, specifies that all control characters received will be link-escape encoded for transparency. By default, only the characters represented by hexadecimal 10, 11, 13, 90, 91, and 93, and the sequence “@-CR” are link-escape encoded.

BLASTscript variable: @ZMCTLESCR

Limit Block Length **[0] 24 – 1024**

Overrides the default block length, which is determined by the Baud Rate of the connection.

<u>Baud Rate</u>	<u>Block Length (in bytes)</u>
300	128
600, 1200	256
2400	512
4800 or greater	1024

Specifying a value between 24 and 1024 limits the block length to the new value. A value of 0 specifies the default block length as determined by the baud rate.

BLASTscript variable: @ZMBLKLN

Limit Frame Length [0] 24 – 1024

For Zmodem transfers, specifies a frame length that forces the sender to wait for a response from the receiver before sending the next frame. The default, 0, specifies no limit to frame length.

BLASTscript variable: @ZMFRMLEN

Size of Tx Window [0] – 9999

Specifies the size of the transmit window, which regulates how many data subpackets can be “outstanding” (unacknowledged) before the sender quits sending and waits for acknowledgements. A value of 0 specifies no limit to window size.

BLASTscript variable: @ZMWINDOW

CRC 16 [32]

Specifies the CRC error-detection method to be used, either 16-bit or 32-bit.

BLASTscript variable: @ZMCRC

Auto Receive YES [NO]

Specifies Auto Receive mode, in which downloading begins immediately after Filetransfer mode is entered.

BLASTscript variable: @ZMAUTODOWN

File Conversion [ASCII] BINARY

Specifies whether received files will be treated as ASCII or BINARY. For correct file conversion to ASCII, the remote computer must send the files as ASCII. This implementation of Zmodem does not perform any special handling of CR/LF and LF sequences; however, when appending an ASCII file, the CTRL Z end of file marker is removed before appending the new text.

BLASTscript variable: @ZMCONVR

File Management

NONE PROTECT
[CLOBBER] APPEND

Specifies a file management option for files received. Possible values are:

NONE – The file is transferred if it does not already exist on the receiving system.

PROTECT – The file is transferred only if it does not already exist on the receiving system even if the sending system has specified CLOBBER.

CLOBBER – The file is transferred whether or not it already exists on the receiving system unless the sending system has specified PROTECT.

APPEND – The file is appended to a file of the same name on the receiving system based on the value of the receiving system's File Conversion setting.

BLASTscript variable: @ZMMANAGR

END OF PROTOCOL SUBWINDOW DESCRIPTIONS

Packet Size

1 – 4085 [256]

For BLAST protocol transfers, specifies the packet size that your system will use when it transfers a file. For LAN operations, it also indicates the size of the packets delivered to the network when using the NETBIOS or IPX drivers for node-to-node file transfers.

In general, larger packets result in faster throughput; however, larger packets will pose problems on a noisy connection. Use larger packet sizes when there is little line noise or you are using V.29 “ping pong” modems.

This field “negotiates” down. The versions of BLAST running on the local computer and the remote computer will compare values and use the smaller of the two values.

While transferring files, watch the line quality and retry count in the upper right part of the screen. If the quality of the line varies, or there are a significant number of retries (more than one retry in 20–50 blocks), a smaller packet size will usually improve throughput. The default value, 256, is the optimum setting for most users.

IMPORTANT: When transferring files with BHOST, always set the Packet Size to at least 200, which is BHOST's minimum packet size.

BLASTscript variable: @PAKTSZ

Chapter 6

BLAST Session Protocol

What is a Protocol?

In the serial communications world, a “protocol” is a set of rules that determines how two computers will communicate with each other. These rules define, for example, how to package data for transfer, how to detect damaged data, and how to optimize throughput. Both computers must use the same protocol for a communications session to succeed.

Simple Protocols

During the early days of telecommunications, people who needed to transfer a file across a phone line or a hardwired asynchronous connection were limited to using text transfer. This is the simplest transfer method, involving only the capturing and transmission of the data stream with no error detection. To receive a file, a buffer is opened to save the information; to send a file, the characters from the chosen file are sent directly out of the communications port to the remote computer.

Of course, no telecommunications connection is perfect, and users soon found that line noise could easily corrupt a file. Thus, file transfer protocols were developed to provide error control. Kermit, Xmodem, Ymodem, Zmodem, and FTP are examples of protocols widely

used by computer owners to transfer files. These transfer protocols are fully described the three chapters following this chapter.

The BLAST Session Protocol

BLAST protocol is BLAST, Inc.'s proprietary session protocol. Under the BLAST session protocol, three kinds of tasks can be performed:

1. *Files can be transferred between local and remote machines.*
The BLAST session protocol permits files to be transferred bi-directionally—that is, data is sent and received at the same time with automatic error detection and data compression.
2. *Files on the remote machine can be manipulated.* For example, files can be deleted, renamed, or printed on the remote computer. Because these tasks are mediated by the BLAST session protocol, the commands cannot be garbled by line noise. In addition, the commands are automatically translated into the appropriate instructions on the remote computer. For example, when you give the “List Files” command using the BLAST session protocol, you will receive a directory listing whether the remote machine is a Macintosh, a VAX, or a computer running the UNIX operating system. You do not need to know the machine-specific instruction.
3. *Messages can be exchanged between the local and remote computer.* Between file transfers, if someone is present at the remote site, you can transmit messages to and from the remote operator.

The BLAST session protocol is more sophisticated than many file transfer protocols and is generally faster than most file transfer protocols because it offers *all* of the following features:

- ◇ Bi-directional transfers.
- ◇ Six levels of compression.
- ◇ Sliding-window design.
- ◇ Automatic translation of text files between the local file format and the format of the remote system.
- ◇ Resumption of interrupted file transfer from the point of interruption.

BLAST Protocol Design

Bi-Directional and Sliding-Window Capability

The BLAST protocol is capable of transmitting and receiving data packets simultaneously. This simultaneous bi-directional transfer saves time and online charges when files need to be both sent and received.

BLAST operates efficiently over circuits with high propagation delays (the length of time from when a character is transmitted to the time it is received). This resistance to delays is due to BLAST's sliding-window design.

The size of a window is the number of packets that can be sent to the remote computer without BLAST's having to wait for an acknowledgement from the remote. As the remote computer sends acknowledgements, the window slides so that more packets can be sent. For example, if the window size is set to 16, and the first 4 of 12 packets sent have been acknowledged, the window slides to allow 8 more packets to be sent. In this way, a continuous stream of packets can be sent without BLAST's having to wait for an acknowledgement. The window size and frequency at which acknowledgements are requested can be specified by the user.

These two features—simultaneous bi-directional transfer and sliding-window design—combine to make BLAST a great time saver for long-distance callers. For example, BLAST can upload daily production figures to a remote computer over a noisy telephone line at the same time that it downloads the next day's production quotas.

CRC Error Detection

BLAST protocol uses the industry-standard CCITT CRC-16 technique for detecting altered data packets. This is the same method used in IBM SNA/SDLC networks and X.25 packet-switching networks.

Optimized Acknowledgements

When packets of data are transmitted, they must be acknowledged by the receiving computer so that the sender knows that the transfer is complete and accurate. When data is being transmitted in only one

direction, the BLAST protocol uses a minimal number of acknowledgement packets flowing in the opposite direction. When data is being transferred in both directions, the data and acknowledgement packets are combined into a single packet. This efficient use of packets is important when working with networks because network charges are often computed on a per-packet rather than a per-byte basis.

Adjustable Packet Size

The BLAST packet size can be set from 1 to 4085 bytes according to the quality and type of connection. A small size minimizes the amount of data that must be retransmitted if line noise is a problem. With high quality connections or with error-detecting modems, packet size can be increased to reduce transmission overhead. Packet size can also be set to optimize network packet use.

BLAST Protocol Circuit Requirements

BLAST is flexible in its circuit requirements. Because BLAST does not use any of the ASCII control codes, it is compatible with the use of these control codes for other purposes. For example, if the XON/XOFF Pacing setup field is set to ON, BLAST can be used on circuits where software flow control (CTRL Q and CTRL S) is in use. This feature is very important for load sharing on network virtual circuits and time-shared mini-computers. In contrast, because they use CTRL Q and CTRL S, Xmodem and Ymodem protocols cannot be used for file transfers with devices that honor XON/XOFF control.

BLAST can operate on 7-bit or 8-bit circuits. 7-bit operation allows BLAST to communicate with parity. This does not inhibit BLAST's ability to transmit binary data—you may transfer either 7- or 8-bit data over both 7- and 8-bit circuits.

When using BLAST to communicate with computers that require 7-bit circuits, the setup parameter 7-Bit Channel must be set to YES. Note that transfer throughput is slower over a 7-bit channel than it is over an 8-bit channel.

Starting a BLAST Session

Starting BLAST on a Multi-User System

There are three ways to start a BLAST session on a remote multi-user computer after your PC connects. Note that you should already be logged into the remote system and appropriate directory.

Manual Method

- ◇ Select Terminal from the Online menu.
- ◇ Type the appropriate commands to the remote computer to start a BLAST session. Usually this is:

```
blast -h
```

at the command line.

- ◇ You should see either one of two messages from the remote:

```
;starting BLAST protocol.
```

or

```
ppp... (only for earlier versions of BLAST)
```

After the message appears, press *ATTN ATTN* to exit Terminal mode; then select Filetransfer from the Online menu.

Interactive Automatic Method

Select Filetransfer from the Online menu. Your PC will automatically start the BLAST session on the remote system.

NOTE: The type of multi-user remote operating system must be identified in the System Type setup field for this method to work. BLAST will then know which automation information to retrieve from the SYSTEMS.SCR library program.

BLASTscript Automatic Method

Write a BLAST script that includes a `FILETRANSFER` statement. This script can be executed from the command line or from the Online menu. The `FILETRANSFER` statement will start a BLAST session on the remote system and initiates the BLAST session locally.

NOTE: The type of multi-user remote operating system must be identified in the System Type setup field for this method to work. BLAST will then know which automation information to retrieve from the SYSTEMS.SCR library program.

Starting BLAST on a PC or Other Single-User Computer

If the remote computer is a single-user system, such as a another PC, you may start the BLAST session in one of three ways:

Assisted Method

- ◇ Select Connect from either the Offline or Online menu.
- ◇ Select Filetransfer from the Online menu.
- ◇ Have the operator on the remote machine select Filetransfer from the BLAST menu.

After the session has started, you can control both PCs from your keyboard; therefore, the remote operator is no longer necessary. In order for you to be able to complete all transfers and end the session without remote assistance, however, the remote operator must press ESC before leaving so that the remote system will terminate the session on your command.

Unattended Method

- ◇ Run the BLAST script SLAVE.SCR on the remote PC. This script places the remote in “slave” mode, waiting for incoming calls. See “Slave Script” on page 198 for details.
- ◇ Select either the Offline or Online menu Connect command.
- ◇ When connected, you have 25 seconds to select Filetransfer from the Online menu. If Filetransfer is not selected within this time, the slave assumes the call is not for BLAST, hangs up the modem, and resets for the next call. When the remote receives your Filetransfer command, it automatically initiates the BLAST session.

BHOST

- ◇ Run BHOST on the remote system if the remote system is a PC running DOS. BHOST occupies less than 100K of RAM and performs file transfers in background mode.
- ◇ After establishing a connection with the BHOST machine (see “Connecting to the Host PC” on page 304), initiate the local

BLAST session by any of the methods described above. BHOST will automatically complete the protocol link.

Automatic Filetransfer Handshaking

While entering Filetransfer mode, the two computers will communicate for a few seconds on their own—they will “shake hands” by exchanging information. During handshaking, your PC will:

- ◇ Send its BLAST version and type to the remote computer to be displayed and written to a log file if it exists.
- ◇ Exchange filetransfer and communication parameters with the remote computer and adjust itself to the other machine’s lowest setup values. For instance, if your setup specifies a Packet Size of 256 bytes and the remote computer is set to 2048, then the lower value of 256 will be used.
- ◇ Display the Filetransfer menu and an initial assessment of communication line quality.

This process can fail if it does not occur within the time period specified in the Logon Timeout setup field. If handshaking fails, BLAST displays a “Logon Timeout” error message and returns to the Online menu.

BLAST Protocol Timeouts

There are two types of timeouts in BLAST protocol: the Logon Timeout and the Inactivity Timeout. Both timeout values can be specified in setup fields of the BLAST Protocol subwindow (see page 87).

The Logon Timeout is the maximum time in seconds after initiating the BLAST session protocol that BLAST will wait for the initial handshake with another system. The default value is 120. If a Logon Timeout exists and the maximum time specified to establish the session elapses, BLAST will return to the Online menu.

Setting the Logon Timeout 0 disables the timeout. If the initial handshaking between computers is aborted and the Logon Timeout on the remote computer is set to 0, the remote computer will attempt to enter BLAST session protocol indefinitely, making the remote computer inaccessible. You may, however, force the remote system to abort its attempt to enter BLAST session protocol by following these steps:

- ◇ Select the Terminal command to enter Terminal mode.
- ◇ When you see the BLAST message

;starting BLAST protocol.

on the display, type:

;DISC.

Because the message you type will not be echoed on the screen, repeat it several times if necessary. Note that the command is case-sensitive. The remote system prompt will appear when the remote computer has aborted its attempt to enter BLAST session protocol.

The Inactivity Timeout is the maximum time in seconds allowed between the transmission of valid BLAST protocol transfer packets. The default is 120 seconds. If BLAST times out, it will return to the Online menu.

Setting the Inactivity Timeout to 0 disables the timeout. If file transfer is interrupted and the Inactivity Timeout on the remote computer is set to 0, the remote computer will remain in Filetransfer mode indefinitely, making the remote computer inaccessible. You may, however, force the remote system to abort Filetransfer mode by entering Terminal mode and typing “;DISC.” as described above (you will probably not, however, see “*;starting BLAST protocol*”).

NOTE: Using the Local menu during a file transfer suspends transfer activity, causing Filetransfer mode to terminate if the Inactivity Timeout interval is exceeded.

Ending a BLAST Session

The BLAST session can be terminated in one of four ways:

Normal Menu Escape

Press ESC at the Filetransfer menu or include an ESC statement in a BLAST script to end a filetransfer session.

- ◇ The files queued for transmission and the files currently being processed complete transmission normally.

- ◇ The computers complete an exit handshake and display normal end messages.
- ◇ Control passes to the Online menu or to the BLASTscript statement following the ESC.

Note that for completion of the handshake, the remote operator must have pressed ESC unless the remote system is in host mode or is running a script with an ESC statement, in which case the remote system will automatically recognize your command.

Single-Attention Abort

Press the *ATTN* key once to quit a transfer:

- ◇ The files queued for transmission will not be sent, and the file currently being transmitted will be marked on the receiving side as interrupted.
- ◇ The computers complete an exit handshake and display normal end messages.
- ◇ Control passes to the Online menu or to the BLAST script.

Double-Attention Abort

Press the *ATTN* key twice to quit immediately.

- ◇ The files queued for transmission will not be sent, and the file currently being transmitted will be marked on the receiving side as interrupted.
- ◇ The computers do not complete an exit handshake.
- ◇ The remote is left to time out on its own. You may force a disconnect by typing ;DISC. as described earlier.
- ◇ Control passes to the Online menu or to the BLAST script.

Timeout Abort

If a communications failure causes a timeout, the phone is disconnected, or no activity takes place, both computers send an exit handshake when the timeout value is reached.

Performing Filetransfer Commands

Filetransfer Menu

After the handshaking is completed, BLAST will display the Transfer Status Area and the Filetransfer menu as shown in Figure 6-1 below:

FIGURE 6-1

```
BLAST Filetransfer hardwire C:\BLAST MENU
Send Get Message Remote Local File
... send file(s) to the remote system
_ local _____ opt = % xfer = file size = ln qual _
S: <idle>
R: <message> good (00)
```

The basic functions of a filetransfer session are controlled by the following commands of the Filetransfer menu:

Send – Sends a file or files to the remote system.

Get – Receives a file or files from the remote system.

Message – Sends a text message of up to 67 characters in length to the remote operator. Simply type the message and press ENTER. The message will be queued for transmission to the remote display following completion of other pending filetransfer commands.

Remote – Takes user to the Remote menu, which performs remote system commands. See “BLAST Protocol Remote Menu” on page 120 for more detailed information.

Local – Takes user to the Local menu, which performs local system commands. See “The Local Menu” on page 57 and the note concerning the Local menu and Inactivity Timeout under the section “BLAST Protocol Timeouts” on page 107.

File – Executes a transfer command file that can control an entire filetransfer session unattended (see “Transfer Command File” on page 117).

Transfer Options

Three transfer options can be used in file transfers via the Filetransfer menu command or a BLASTscript FILETRANSFER statement:

- t translate the file from the originating system’s text file format to the destination system’s text file format. This option should

only be used with ASCII files—do *not* send binary files using the `t` option.

- have the transmitted file overwrite an existing file with the same name on the receiving system. This option will result in the destruction of the original file on the receiving system, so use it with caution. An error will result if this option is not used and the file already exists on the receiving system.
- a append the transmitted file to the end of an existing file with the same name on the receiving system. If the file does not exist on the receiving system, it will be created.

When using the Filetransfer menu command, you are prompted to type one or more of these letters (`t`, `o`, or `a`) to specify your transfer option(s). In a BLAST script, type the letter(s) on a separate line following the name of the file or files to be transferred. For more on using transfer options in a BLAST script, see “Getting and Sending Files” on page 178.

Sending a File

To send a file,

◇ First, select Send from the Filetransfer menu by pressing `s`.

◇ At the prompt:

enter local filename:

enter a single filename from the current directory or a path specification with a single filename; you may use wildcards and file transfer switches (see the section “Wildcards” on the next page and “File Transfer Switches” on page 114). After entering the filename and any switches, press ENTER.

◇ At the prompt:

enter remote filename:

Press ENTER or type a single filename with any optional switches or type a “%”.

Pressing ENTER only will transfer the file to the remote system using the local filename (and path if included with the local filename). Typing a different filename (and path, if necessary) will rename the file when it is created on the remote system. See

“File Transfer Templates Using the ‘%’ Character” on page 113 for an explanation of “%”.

IMPORTANT: If wildcards are used in the local filename designation, you must use “%” when sending from a PC to another system type, such as a UNIX machine.

Some remote computers will interpret optional file transfer switches sent with the remote filename as file-handling and file-attribute controls. If you have specified a remote filename, press ENTER.

◇ At the prompt:

specify transfer options: (t=text, o=overwrite, a=append):

type any combination of the letters t, o, and a or press ENTER only to specify no options. For a fuller description of transfer options, see the preceding section, “Transfer Options.”

If you do not specify any options, the file will be transferred to the remote system byte-for-byte as a binary file. If the file exists on the remote system, the transfer will abort.

After specifying options, press ENTER; you will be returned to the Filetransfer menu, and the transfer will begin. The number of bytes sent will appear, as well as a percentage estimate of the amount of data transferred. When the file transfer completes, a message will be sent to your system.

Getting a File

Receiving a file differs only slightly from sending a file. Press G from the Filetransfer menu. You will be prompted for the remote filename first and then the local filename. Any switches added to the end of the remote filename must be valid for that operating system.

Wildcards

By using the wildcard characters “*” and “?”, you can transfer multiple source files with similar names. The source files must reside in the same directory and path. The wildcard specifications are as follows:

? Substitutes for a single character.

- * Substitutes for multiple characters (up to 8 characters of the filename, up to 3 of the extension).

File Transfer Templates Using the “%” Character

When a “%” is entered in the filename field for the target drive, filename(s) from the source drive are transferred to the target drive without the source drive path specification(s).

IMPORTANT: “%” is *required* for the target filename when the source filename contains a “?” or an “*” or when the source filename includes a path and the target filename does not (that is, the target directory is the current working directory).

Some examples are:

<u>Source Name</u>	<u>Target Name</u>	<u>Result</u>
C:\test1.asc	C:\test1.asc	one file in directory C:\ sent to the target directory C:\ (C:\ must exist).
C:\test1.asc	%	one file in directory C:\ sent to the remote current directory
C:\test?.asc	%	multiple files—for example, test1.asc, test2.asc, and test3.asc—in directory C:\ sent to the current target directory, retaining their source names
C:\test1.*	%	multiple files in directory C:\—for example, test1.asc, test1.lst, and test1.txt—sent to the current target directory, retaining their source names
C:*.*	C:\BIN\%	all files in directory C:\ sent to the target directory C:\BIN\, retaining their source names.

File Transfer Switches

Instead of specifying transfer options at the prompt, you can append the appropriate file transfer switches to both the local and remote filename specifications. Some remote computers will recognize switches sent with the remote filename as file-handling and file-attribute controls. Experiment with the transfer switches until you obtain the correct results. The valid switches are:

- /APP** Append to a file with the same name if it exists.
- /COMP=*n*** Switch compression level value from the value in the compression field of the setup. Use the **/COMP=*n*** switch at the end of the filename where *n* equals the level of compression (0–6). Setting the level to 0 turns off compression.
- /FWD** Delete file from sending system if the transfer was successful. The **/FWD** switch is disabled by default. To enable it, toggle the Enable **/FWD** and **/STR** setup field (page 91) in the BLAST Protocol subwindow to YES. For the **/FWD** switch to work, it must be enabled on the *sending* system.
- NOTE:** The **/FWD** switch is a very powerful feature of BLAST. Because it allows files to be automatically *deleted* from the sending system, always exercise caution when using it.
- /OVW** Overwrite a file with the same name if it exists. The ability to use the **/OVW** switch is enabled by default. To disable use of it, toggle the Enable **/OVW** and Remote Cmds setup field (page 91) in the BLAST Protocol subwindow to NO (see also “Disabling File Overwrites and Remote Commands” on page 123).
- NOTE:** If use of the **/OVW** switch is disabled on the receiving system, BLAST protocol will not allow the file to be overwritten.
- /STR** Delete file from receiving system if transfer was unsuccessful. The **/STR** switch is disabled by default. To enable it, toggle the Enable **/FWD** and **/STR** setup field (page 91) in the BLAST Protocol subwindow to YES on the receiving system (see

also “Disabling the /FWD and /STR Switches” on page 123).

/TXT Perform text translation. BLAST will convert carriage returns, line feeds, and end-of-file markers to the receiving system’s text format.

You might, for example, specify text translation and overwriting of an existing file with the following filename:

```
test1.doc/TXT/OVW
```

Or you might specify that the file be sent with a compression level of 6 and automatically deleted from your system if it is successfully sent:

```
test1.doc/FWD/COMP=6
```

Filenames Restrictions with BLAST Protocol

With BLAST protocol, you should *not* give a file the same name as a switch since BLAST protocol will assume that the file is a switch and will look for a file with the name of the folder containing the file if forward slashes are used in the pathname. Thus, the transfer of the file will not occur and you will get an error message. Some of the more common switches to avoid in naming files (uppercase or lowercase) are: app, comp=*n*, follow=*nm*, fwd, group, ovw, owner=*nm*, perms=*nnnn*, str, and txt (where *n* is a number from 0 to 9).

You can work around this restriction by using backslashes for the local path and, if the operating system allows, the remote path as well. If the remote path does not allow backward slashes, you can change your remote working directory to the one containing the file you want to transfer and use % instead of the remote path and filename (see the discussion of % on page 113). To change your remote working directory interactively, choose the Chdir command from the Remote menu. Alternatively, you may do a scripting workaround:

```
FILETRANSFER
REMOTE
  Chdir                    # Change remote working directory
  /u/customer            # Name of new directory
  ESC
SEND
C:\Pat\Work\App        # Local path with backslashes
%                        # Original filename w/ path stripped
ESC
```

If, on the receiving system, you give the file a new name that is not that of a switch, you *can* give a path. For instance, if in the script above, App was given the new name sales.txt on the receiving machine, you could change the script to the following:

```
FILETRANSFER
```

```
Send
```

```
C:\Pat\Work\App          # Local path with backslashes  
/u/customer/sales.txt    # New filename and full path
```

```
Esc
```

Restarting an Interrupted File Transfer

Disconnections and interruptions in sending long files can be costly and time-consuming. BLAST can restart transfer of files from the point of interruption without having to restart transmission from the beginning of the file.

If a filetransfer session is interrupted and you wish to restart from the point of interruption, both local and remote systems must time out or be interrupted by *ATTN ATTN*. After the session has been interrupted or aborted, you may restart the session by following these steps:

- ◇ Reconnect, if necessary, and restart the filetransfer session.
- ◇ Send the *exact* file that was being sent when interrupted.
- ◇ Do *not* indicate the overwrite or append options.

BLAST restarts from the last point at which its buffers were flushed to disk. This may be right at the interrupt point or as much as 10K before the interrupt point.

NOTE: Adding the */STR* switch to a filename eliminates the possibility of resuming an interrupted transfer of that file.

Transfer to a Printer

With BLAST protocol, you may specify a printer as the destination on a remote computer. For instance, in a PC-to-PC transfer, the Sending PC would use the device name “PRN” as the destination filename on the receiving PC. BLAST will exert flow control as necessary to accommodate the speed of the printer, but you should be aware that unexpected events, such as the printer running out of paper, will cause BLAST to time out and end the transfer.

NOTE: If you have the disk space available, it is always better to send a file to disk and then print the file.

When you send to a printer, restarting from the point of interruption is not supported; also, you may need to specify the /TXT switch for the data to print properly.

An example in BLASTscript might look like this:

```
FILETRANSFER
SEND
sample.fil
PRN
T
ESC
```

Transfer Command File

A transfer command file is a text file that contains line-by-line instructions describing functions to be performed during a BLAST protocol filetransfer session. Any word processor or editor can create a transfer command file, but it must be saved in text only or ASCII format under any name that you choose. Transfer command files are also called error-free command files.

A transfer command file can be invoked interactively by selecting the File command from the Filetransfer menu, or from within a BLAST script by using the following BLASTscript commands:

```
FILETRANSFER
FILE
Filename    # name of the transfer command file
ESC
```

If the transfer command file is in the current directory, you only have to specify the filename; if it is in any other directory, you must specify the full path.

The command file contains an unlimited number of commands, each as a separate line of text. Files, messages, and certain remote commands can be sent and remote files can be received. Filetransfer commands are entered as one line, with the source and destination specifiers separated by a space. If any file transfer switches are required, they are entered following the file specifier(s).

Command Formats

The text in a transfer command file must begin in the first column of every line. Commands in a transfer command file accomplish one of four tasks:

1. Send a File:

No special character is required; simply type the name of the local file to send and, separated by a space, the name for the file on the remote system. If no remote name is given, BLAST will use the local name. Any file transfer switches must be typed immediately following the filename:

```
local_filename[switches] [remote_filename[switches]]
```

2. Get a File:

The first character in the line must be a plus sign (+). Immediately following the “+”, enter the name of the file to receive from the remote system and, with no intervening space, any file transfer switches. If a different name is desired for the local file, type a single space after the remote filename and then type the local filename with any switches immediately following:

```
+remote_filename[switches] [local_filename[switches]]
```

Note that it is more efficient to put all Gets (lines beginning with “+”) first, so that the remote file requests queue up on the remote. This order allows for true bi-directional transfer during command file operations.

3. Send a Display Message:

The first character in the line must be a semicolon (;). Immediately following the semicolon (;), type the desired message, which will be transmitted to the remote display and the remote log, if specified:

```
;Now Sending Sales Reports
```

4. Send a Command to the Remote Computer:

The character in the first column must be an exclamation point (!). Immediately following the exclamation point, type the command to be sent to the remote computer, for example:

```
!dir
```

The valid remote commands are:

DIR

Display the contents of the current remote directory.

TYPE *filename*

Type the contents of the specified remote file to the screen.

C

Display the next page of a multi-page display.

PRINT *filename*

Print the specified file on the remote printer.

REN *oldname newname*

Rename the specified remote file to the new name.

ERA *filename*

Erase the specified remote file.

CHDIR *path*

Change from the current remote directory to the specified remote directory.

Example

To understand the use of transfer command files, imagine that a salesman named Joe is using BLAST to keep track of current pricing changes and to send in current orders. He will always get the file called CURPRICE.FIL and send the file called JOEORDER.FIL. Joe can create an error-free command file named JOE.CMD, which looks like this:

```
;I want to get current price lists
+CURPRICE.FIL/txt JOEPRICE.FIL/txt/ovw
;Now I am about to send in today's orders
JOEORDER.FIL/txt TODAYORD.FIL/txt/ovw
!DIR
```

To use this command file, Joe would choose File from the Filetransfer menu and type in the name JOE.CMD at the prompt. The following sequence of events then takes place:

- ◇ The first message in the command file appears on the screen.
- ◇ The file CURPRICE.FIL is retrieved and overwrites the old JOEPRICE.FIL.
- ◇ The second message appears.
- ◇ JOEORDER.FIL is sent and overwrites the old TODAYORD.FIL.
- ◇ Finally, the contents of the current directory of the remote computer are displayed on Joe's screen.

BLAST Protocol Remote Menu

The Filetransfer menu contains a Remote command that takes you to the Remote menu. The Remote menu allows a user with no knowledge of the remote operating system to manage files on that system. For example, you can delete a file on a remote UNIX system without actually typing the UNIX delete command. BLAST will “translate” the command automatically. Remote commands affect only files in the current remote directory unless you specify a path-name.

NOTE: The Enable /OVW and Remote Cmds setup field (page 91) in the BLAST protocol subwindow must be enabled on the remote system in order for you to delete, rename, or print files on the remote system.

Following is a description of the Remote menu commands:

List – Operates like the Local List command except that it displays the contents of the current remote directory. You will be prompted to choose either a detailed (long) or non-detailed (short) list and then to specify a filename; you may use a specific filename, a filename with wildcard characters (for example, “*”), or press ENTER to display all files in the current remote directory.

Delete – Deletes a single file or multiple files from the remote system. You may use a specific filename or a filename with wildcard characters (for example, “*”).

Rename – Renames a remote file.

Type – Displays a remote file on the BLAST screen.

Print – Prints a remote file to the remote printer.

Chdir – Changes the current remote directory to one that you name. BLAST will check this directory for any files that you specify with the Remote menu commands.

More – Scrolls a page of data when either the List or Type commands causes more than one full screen of data to be received. You will be prompted to execute the More command in order to see the remaining pages, one page at a time.

Automating the BLAST Session Protocol

The BLAST session protocol can be fully automated through scripting. For information on writing scripts using the BLAST protocol, see “File Transfers with BLAST Session Protocol” on page 178.

Fine-Tuning the BLAST Session Protocol

You may fine-tune throughput by adjusting packet size and compression level:

Packet Size

Most computers can process packets of 256 characters. Set the Packet Size setup field (page 99) to 256 or higher unless phone line quality is poor. Small packet sizes reduce the number of bytes requiring retransmission over noisy lines. Computers connected directly by cables will benefit from a much larger packet size, such as 4085. In a BLAST script, the reserved variable for packet size, @PAKTSZ, can be set anytime before entering a filetransfer session.

Compression Levels

BLAST performs automatic data compression during file transfers with the BLAST protocol, reducing the number of characters sent and the transfer time.

Compression level is specified in BLAST Protocol subwindow setup fields (page 91). Possible values for Receive Compression Level

and Send Compression Level are 0 (no compression) to 6. The default is 4, which provides the best performance for average-sized files. Compression can also be selected by the @RCOMP_LEV (receive) and @SCOMP_LEV (send) BLASTscript reserved variables.

Data compression requires additional RAM during file transfers. The amount of RAM necessary varies with the compression level. Following is a brief explanation of each compression level and recommendations for its use:

Compression Level 0 – Level 0 specifies that no compression will be used. Choose level 0 when your CPU is slow and the baud rate is high. In this situation, the overhead needed for compression can actually increase transfer time.

IMPORTANT: Always use compression level 0 when transferring pre-compressed files.

Compression Level 1 – Use level 1 when your data has strings of duplicate characters. Such data could include row and column reports, which have many embedded blanks, and executable files with blocks of nulls. In some cases, compression level 1 improves performance over high-speed modems with hardware data compression enabled.

Compression Level 2 – Starting with level 2, compression requires more work by both computers. With a standard modem and two fast machines, however, levels 2–4 will save transmission time.

Compression Level 3 and 4 – Levels 3 and 4 of compression are most effective when a limited character set is used or there are repetitious patterns. Because spreadsheets and databases have many repetitious patterns and a limited character set, they are highly compressible.

Compression Level 5 and 6 – Levels 5 and 6 compression are most effective for large files (above 500 K). On large files, the receiving computer may notice a significant delay before the first block is received while the sending computer calculates maximum compression.

Disabling File Overwrites and Remote Commands

The Enable /OVW and Remote Cmds setup field (page 91) and the BLASTscript variable @ENABLERCMD (page 248) control whether or not remote commands and file overwrites are allowed during Filetransfer mode. Note that disabling /OVW affects only local files. For example, you will still be able to send a file with the /OVW switch because the file will be overwritten on the remote system.

Disabling the /FWD and /STR Switches

The Enable /FWD and /STR setup field (page 91) and the @ENABLEFS (page 248) script variable control whether or not the /FWD and /STR file transfer switches are allowed during Filetransfer mode. Note that disabling these switches affects only local files. For example, you will still be able to get a file with the /FWD switch because the transferred file will be deleted from the remote system. See “File Transfer Switches” on page 114.

NOTE: Adding the /STR switch to a filename eliminates the possibility of resuming an interrupted transfer of that file.

Using the Transfer Password

When you specify a Transfer Password on your PC, BLAST will restrict access by a remote user during BLAST protocol transfer. If the remote user does not have the password, he may only send and receive messages while in Filetransfer mode. The Transfer Password can be set by entering it into the Transfer Password setup field (page 88) or by setting the reserved variable @TRPASSWD (page 269) in a slave script on the remote system.

After entering a filetransfer session, the remote user must send the transfer password to your machine using the Send command from the Filetransfer menu or a FILETRANSFER statement in a script. If the user issues a Send command from the Filetransfer menu, the following special format for the local filename must be used:

```
!password=your_password
```

where *your_password* represents the password stored on the your system. The remote filename field is left blank as are the text, overwrite, and append options. If the correct password is successful-

ly sent, the remote user will see a message stating that the password has been validated. *The password must be typed exactly as it is set on the your system.*

If a BLAST script is used, the same special local filename format must be used for sending the password to your computer, for example:

```
FILETRANSFER
SEND
!password=blue2
```

```
SEND
myfile.rpt
yourfile.rpt
ta
ESC
```

Because the remote filename and send transfer options are not used, two blank lines must follow the `!password=your_password` statement. See “Getting and Sending Files” on page 178 for information on scripting file transfers.

Since the remote user has to enter the password through BLAST interactively or through a script, the use of Transfer Password deters an unauthorized user from breaking your security by submitting a rapid series of passwords.

NOTE: The Transfer Password is intended to validate remote users logging onto your system. If a local operator uses a setup with a Transfer Password entered, he or she will not be able to receive files without the remote user sending the password.

Chapter 7

FTP

With a compatible packet driver installed, BLAST can establish FTP connections over TCP networks using BLASTTCP (see “TCP/IP Driver” on page 33 for information on installing and configuring BLASTTCP). For information on scripting FTP sessions, see “File Transfers with FTP” on page 181.

Using FTP

Most users will be transferring files to a multi-user host such as a UNIX-based computer. Create a setup that includes the following choices:

```
System Type: UNIX
Connection: TCP/IP ->
Modem Type: None
Originate/Answer: Originate
Protocol: FTP
```

Be sure you have also entered your Userid and Password so that BLAST can log you into the remote system. The Connection field should contain the name of your local server.

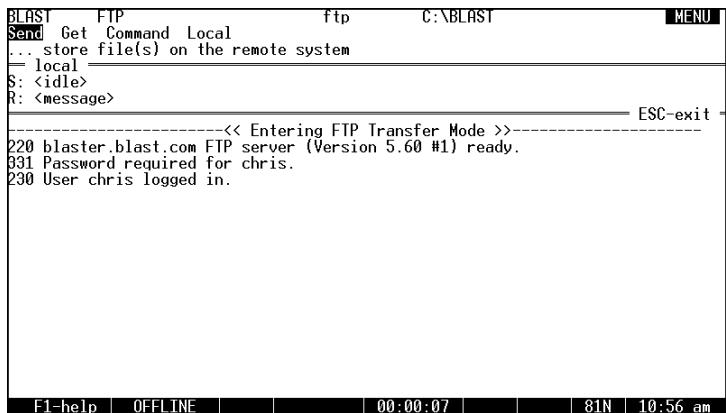
Starting an FTP Session

After creating the appropriate setup file, you can establish FTP sessions in two ways. If you use the Connect command first from the Offline menu, FTP will start a Telnet session with the remote host. You can then issue the Filetransfer command, which will suspend the Telnet session and start the FTP session. When you exit FTP, your Telnet session will resume.

Alternatively, you can go to the Online menu and issue the Filetransfer command without connecting first. Blast will establish the FTP session and close the connection after you leave filetransfer.

FTP Filetransfer Menu

The FTP Filetransfer menu (Figure 7-1) is slightly different from the menu displayed during a BLAST protocol session.



```
BLAST  FTP          ftp          C:\BLAST          MENU
Send  Get  Command Local
... store file(s) on the remote system
----- local
S: <idle>
R: <message>
-----<< Entering FTP Transfer Mode >>----- ESC-exit
220 blaster.blast.com FTP server (Version 5.60 #1) ready.
331 Password required for chris.
230 User chris logged in.

F1-help  OFFLINE  |  |  | 00:00:07  |  | 81N  10:56 am
```

FIGURE 7-1

Following is a brief description of the commands of FTP Filetransfer menu:

Send – Sends a file to the an FTP server. You will be prompted for a local and remote filename.

Get – Receives a file from an FTP server. You will be prompted for a remote and local filename.

Command – Sends a command to the remote computer. You will be prompted for the command. For example, if you

type “help” and press ENTER, you will see a list of commands available on the remote system. For information on a specific command, type “help” followed by the name of the command, and then press ENTER. See also “FTP Commands” on page 129.

Local – Takes you to the local menu, which performs local system commands (see “The Local Menu” on page 57). Note that all filetransfer activity is suspended while you are using the local system.

Sending and Receiving Files with FTP

The following two sections describe interactive file transfers. For a discussion of scripting FTP file transfers, see “File Transfers with FTP” on page 181.

Sending Files with FTP

To send a file, select Send from the Filetransfer menu and enter the name of the file that you wish to send. You will then be prompted for a remote filename.

IMPORTANT: FTP will overwrite a file of the same name on the remote system without warning you first.

You may use standard wildcard characters in the filenames. If you wish to append a local file to a file of the same name on the remote system, select Command from the file menu and enter APPE followed by a space and the name of the file that you wish to append. This will automatically append the local file to the remote file. For more on FTP commands, see “FTP Commands” on page 129.

When the FTP transfer completes, a message will be sent to your system and you will be returned to the Filetransfer menu.

Getting Files with FTP

Select Get from the filetransfer menu, enter the name of the remote filename, and then the local filename. By default, the file will be stored in the current local directory, but you may specify a specific directory path. You may use standard wildcard characters in the filenames.

NOTE: FTP GETs should be used with caution. In the FTP protocol, the markers for end-of-file and for close-connection are the same. Thus, incomplete file receives resulting from connection failures are reported as successful file transfers in both the File Transfer Status Area and the log file.

Filenames Restrictions with FTP

With FTP, you should *not* give a file the same name as a switch since BLAST FTP will assume that the file is a switch and ignore it if forward slashes are used in the pathname. In such a case, the transfer of the file will not occur, and you will get an error message. Some of the more common switches to avoid in naming files (uppercase or lowercase) are: `app`, `comp=n`, `follow=nn`, `fwd`, `group`, `ovw`, `owner=nn`, `perms=nnnn`, `str`, and `txt` (where *n* is a number from 0 to 9).

You can work around this restriction by using backslashes if both the local and remote operating systems support them. If not, you can change your local and remote working directories to the ones containing the file you want to transfer and give the local and remote filenames without a path. To change your local working directory interactively, choose `Chdir` command from the Local menu. To change your remote working directory interactively, choose `Command` from the Filetransfer menu and type in `CWD` and the name of the new directory (see “FTP Commands” on page 129).

Alternatively, you may do a scripting workaround:

```
FILETRANSFER
LCHDIR "/u/Pat/work" # Change local dir.
REMOTE
  Cwd                # Change working dir.
  /usr/customer      # Name of new directory
  ESC
SEND
App                  # Filename only--no path
ESC
```

If, on the receiving system, you give the file a new name that is not that of a switch, you *can* give a path. For instance, if in the script above, `App` was given the new name `sales.txt` on the receiving machine, you could change the script to the following:

```

FILETRANSFER
LCHDIR "/u/Pat/work"      # Change local directory
Send
App                        # Filename only--no path
/usr/customer/sales.txt # Give new name and full path
Esc

```

Ending an FTP Session

FTP sessions end automatically when all specified files are transferred or remote commands executed and you press ESC to exit File-transfer mode.

FTP Commands

You can perform commands on the remote computer by choosing Command from the FTP Filetransfer menu and typing one of the following commands at the prompt.

Service Commands

ABOR	Abort last command
ALLO <i>number_bytes</i>	Allocate file space
APPE <i>remote_filename</i>	Append to file
DELE <i>filename</i>	Delete file
HELP [<i>command</i>]	Help
LIST [<i>pathname</i>]	List files
MKD <i>new_directory_name</i>	Make directory
NLST [<i>pathname</i>]	Name list
NOOP	No operation
PWD	Print working directory
REST <i>byte_number</i>	Restart transfer at
RETR <i>remote_filename</i>	Retrieve file
RMD <i>directory_name</i>	Remove directory
RNFR <i>current_filename</i>	Rename from
RNTO <i>new_filename</i>	Rename to
SITE <i>site_parameter</i>	Site specific parliamentary
STAT [<i>pathname</i>]	Return server status
STOR <i>local_filename</i>	Store file
STOU <i>local_filename</i>	Store as unique name
SYST	Return server OS type

Access Control Commands

ABOR	Abort last command
ACCT <i>account</i>	Specify account
CDUP	Change to parent directory
CWD <i>new_directory</i>	Change directory
PASS <i>password</i>	Password
QUIT	Quit transfer and log out
REIN	Reinitialize
SMNT <i>pathname</i>	Structure mount
USER <i>username</i>	Login ID

Transfer Parameter Commands

MODE <i>mode_code</i>	Transfer mode (S or B)
PASV	Passive transfer
STRU <i>structure_code</i>	File structure (F)
TYPE <i>type_code</i>	File type (I or A N)

Chapter 8

Kermit Protocol

Many communication products support Kermit protocol on a wide range of computers, but you should be aware that there are many different versions of Kermit, two with which BLAST can communicate. The simplest version is a file transfer program that requires commands to be entered at both the sending and receiving computers (using the Send and Receive commands). The second, more sophisticated version is the Kermit server. The Kermit server accepts commands from a remote user and performs specified operations (using the Send, Get, and Remote commands). For information on scripting Kermit sessions, see “File Transfers with Kermit” on page 181.

Kermit Filetransfer Menu

You will notice that the Kermit Filetransfer menu (Figure 8-1, next page) is slightly different from the menu displayed during a BLAST protocol session. Below is a brief description of the command options of this menu:

Send – Sends a file to a Kermit program. You will be prompted for the local and remote filenames.

```

BLAST Kermit DEFAULT C:\BLAST MENU
Send Get Receive reMote Finish Bye
... send file(s) to the remote system
== local _____ % xfer = file size = byte cnt = retries ==
S: <idle>
R: <idle>
-----ESC-exit-----
<< Entering KERMIT Transfer Mode >>-----
F1-help | DCD | CTS | | 00:00:34 | | 81N | 01:26 pm

```

FIGURE 8-1

- Get** – Receives a file from a Kermit server. You will be prompted for the remote and local filenames.
- Receive** – Receives a file from a simple Kermit. You must specify a local filename.
- Remote** – Takes you to the Remote menu, which performs remote Kermit server commands. This option allows a user with no specific knowledge of the remote operating system to manage files on the remote computer. For example, a PC user can delete a file without actually typing the delete command of the remote operating system (see “Kermit Remote Menu” on page 135).
- Finish** – Returns you to the Online menu. Kermit server finishes transfer and exits without logging off; thus, you may continue the session.
- Bye** – Ends Kermit server mode and logs off of the remote system. Depending on the remote modem settings, the connection may or may not be broken. You will be returned to the Online menu.

Sending and Receiving Files with Kermit

The following two sections describe interactive file transfers. For a discussion of scripting Kermit file transfers, see “File Transfers with Kermit” on page 181.

Sending Files with Kermit

Kermit Server

- ◇ In Terminal mode, begin the Kermit program on the remote system.
- ◇ Exit Terminal mode, select the Filetransfer command from the Online menu, and then select the Send command. You will be prompted for the local and remote filenames. For the local filename, you may enter a single filename from the current directory or a path specification with a single filename. You may use wildcards (see “Wildcards” on page 112), but you *cannot* use file transfer switches.
- ◇ The transfer will begin, and the number of bytes sent will be displayed in the File Transfer Status Area.

Simple Kermit

- ◇ In Terminal mode, begin the simple Kermit program on the remote system.
- ◇ In simple Kermit on the remote system, issue a receive command.
- ◇ Exit Terminal mode, select Filetransfer, and then select Send. You will be prompted for local and remote filenames. If you designate a remote filename with the simple Kermit receive command, a filename entered at the remote filename prompt will be ignored.

Receiving Files with Kermit

BLAST’s implementation of Kermit supports both the Kermit server Get command and the simple Kermit Receive command to transfer files from a remote computer. Following are directions for transfers from a remote computer:

Kermit Server

- ◇ In Terminal mode, begin the Kermit server program on the remote system.
- ◇ Exit Terminal mode, select the Filetransfer command from the Online menu, and then select the Get command. You will first be prompted for the remote filename—you may enter a single filename from the current directory or a path specification with a single filename; you may include wildcards (see “Wildcards”

on page 112). You will then be prompted for a local filename. Optionally, you may add any supported file transfer switches (see “File Transfer Switches with Kermit” on page 134). Once you have entered the filenames and any switches, the transfer request is automatically sent to the remote.

- ◇ Unless you specify otherwise, the received file will be saved to your current directory.

NOTE: If you have an existing file with the same name, the transferred file will be renamed by adding a number (0001, 0002, etc.) to the original filename when the Warning setup field (page 95) is set to ON. When this field is set to OFF, the existing file will be automatically overwritten.

Simple Kermit

- ◇ In Terminal mode, begin the simple Kermit program on the remote system.
- ◇ In Kermit on the remote system, send the file by invoking the send command.
- ◇ Exit Terminal mode, select Filetransfer, and then select Receive. You will then be prompted for a local filename; optionally, you may add any supported file transfer switches (see the next section “File Transfer Switches with Kermit”).
- ◇ Unless you specify otherwise, the received file will be saved to your current directory.

NOTE: If you have an existing file with the same name, the transferred file will be renamed by adding a number (0001, 0002, etc.) to the original filename when the Warning setup field (page 95) is set to ON. When this field is set to OFF, the existing file will be automatically overwritten.

File Transfer Switches with Kermit

Kermit ignores all file transfer switches on sending filenames and supports the following file transfer switches on receiving filenames:

- | | |
|------|---|
| /APP | Append to a file with the same name if it exists. |
| /OVW | Overwrite a file with the same name if it exists. |

Filenames Restrictions with Kermit

With Kermit Protocol, you should *not* give a file the same name as a switch since, if forward slashes are used in the pathname, BLAST will assume that the file is a switch and either ignore it (if the switch is unsupported by Kermit) or look for a file with the name of the folder containing the file (if the switch is supported by Kermit). In either case, the transfer of the file will not occur and you will get an error message. Some of the more common switches to avoid in naming files (uppercase or lowercase) are: `app`, `comp=n`, `follow=nn`, `fwd`, `group`, `ovw`, `owner=nn`, `perms=nnnn`, `str`, and `txt` (where *n* is a number from 0 to 9).

You can work around this restriction by using backslashes if both the local and remote operating systems support them. If not, you can change your local and remote working directories to the ones containing the file you want to transfer and give the local and remote filenames without a path. To change your local working directory interactively, choose the `Chdir` command of the Local menu. To change your remote directory interactively using Kermit server, choose `Remote` from the Kermit Filetransfer menu and then select the `Cwd` (Change Working Directory) command. To change your remote directory interactively using simple Kermit, access Terminal mode and give the “change current directory” command for that operating system.

Alternatively, you may do a scripting workaround. To change the local working directory, use the `LCHDIR` command. To change the remote working directory using the Kermit server, issue a `FILETRANSFER/REMOTE/Cwd` multi-line command statement. To change the remote working directory using simple Kermit or Kermit server, `TSEND` a “change working directory” command to the remote. For example, the following script fragment changes the current remote directory on a UNIX machine to `/u/sales`.

```
TSEND "cd /u/sales", LF
```

See “File Transfers with Kermit” on page 181 for more on scripting for Kermit.

Kermit Remote Menu

Notice that the Kermit Remote menu (Figure 8-2, next page) offers a selection of commands different than those of the BLAST proto-

col. These functions operate on the remote system in Kermit server mode. However, unreliable results can occur if you use a command that is not supported by the server.

```

BLAST KermitRemote          DEFAULT  C:\BLAST          MENU
Directory Erase Type Cmd Space Who Message h0st Kermit Help
... list remote filenames
----- % xfer = file size == byte cnt = retries -----
S: <idle>
R: <idle>
----- ESC-exit -----
-----<< Entering KERMIT Transfer Mode >>-----
F1-help | DCD | CTS | | | 00:00:34 | | 81N | 01:26 pm

```

FIGURE 8-2

The Remote menu commands are:

Directory – Displays the server’s current working directory or a directory you specify; wildcards can be used.

Erase – Deletes a file in the server’s current working directory or in a directory you specify by giving the full path of the file; wildcards can be used.

Type – Displays a remote file on your local screen. Since Kermit does not support a page pause, you must use CTRL S to pause and CTRLQ to resume the flow of text.

Cwd – Changes the server’s working directory. You will be prompted for the new directory pathname.

Space – Displays the server’s free drive space.

Who – Displays users currently logged onto the remote. If you specify a user name, information on that name only will appear.

Message – Sends a one-line message to be displayed to the remote operator.

Host – Sends an operating system command to the Kermit server. The command is executed immediately.

Kermit – Sends a Kermit language command to modify session parameters, for example, `SET FILE TYPE BINARY`.

Help – Displays a short list of the commands currently available on the Kermit server. Because servers can support different commands, the Help command can be a valuable reminder of what is available through the Kermit server.

The Kermit `DISABLE` command can lock most of these menu commands. For example, the command `DISABLE ERASE` will prevent files from being deleted on the remote system.

Chapter 9

Xmodem, Ymodem, and Zmodem Protocols

BLAST includes Xmodem, Ymodem, and Zmodem protocols for transferring files as an alternative to BLAST protocol.

Before choosing Xmodem, Ymodem, or Zmodem for a major application, ask yourself:

- ◇ Will you need to transfer files with computers using other operating systems?
- ◇ Do your transfers need to be fast and 100% error free?
- ◇ Do you want the ability to execute commands on the remote system without special knowledge of the command syntax?

If you have answered “Yes” to any of these questions, you should use BLAST protocol on your remote system if it is available; Xmodem, Ymodem, and Zmodem protocols do not support both near-transparent remote access to other operating systems nor fast, 100% error-free transfers.

The following instructions are very general. Actual procedures for using Xmodem, Ymodem, and Zmodem will vary depending on the

implementation of these protocols on the remote system. Many communications products support the standard implementation of these protocols; nevertheless, you should be aware that there are different, incompatible versions that might not work successfully with BLAST.

Command Line Features

If you have chosen the Xmodem or Ymodem protocol in your setup, you can specify an end-of-transmission (EOT) timeout parameter using a command line switch in the following format:

```
/tx
```

where *timeout* is equal to *number*/100 seconds. The minimum timeout is .1 second (10), and the maximum is 60 seconds (6000). For example, /t1111 sets the timeout to 11.11 seconds.

You can also select the pad character for Xmodem using the following format:

```
blast /px
```

where *x* specifies the character expressed as a hexadecimal value. For example, /p21 specifies “21” as the pad character.

Invoking a command line parameter affects these protocols only for the duration of that communications session.

Xmodem Protocol

BLAST supports Xmodem1K CRC as well as Xmodem CRC and the standard Xmodem checksum protocol. When you select Xmodem as your protocol, BLAST will automatically determine which implementation of Xmodem is on the remote system and choose the correct counterpart on your local system.

NOTE: Xmodem is only compatible with 8-bit connections.

The following two sections describe interactive file transfers. For a discussion of scripting Xmodem file transfers, see “File Transfers with Xmodem and Xmodem1K” on page 184.

Sending Files with Xmodem

To send a file using Xmodem:

- ◇ Begin the Xmodem receive program on the remote computer, specifying a filename if needed. For example, if the remote computer were a UNIX system running Xmodem, you would enter Terminal mode, begin the remote receive command on the remote computer, and then exit Terminal mode.
- ◇ Select the Filetransfer command from the Online menu, and then select the Send command. You will be prompted for the local filename.

Receiving Files with Xmodem

To receive a file using Xmodem:

- ◇ Begin the Xmodem send program on the remote computer. For example, if the remote computer were a UNIX system running Xmodem, you would enter Terminal mode, begin the remote send command on the remote computer, and then exit Terminal mode.
- ◇ Select the Filetransfer command from the Online menu, and then select the Get command. You will be prompted for the filename. If the file already exists on the local machine, it will be automatically overwritten.

Ymodem Protocol

BLAST supports the standard Ymodem and Ymodem G protocols. *Do not use Ymodem G protocol unless there are properly configured error-correcting modems on both ends of the connection.*

The following two sections describe interactive file transfers. For a discussion of scripting Ymodem file transfers, see “File Transfers with Ymodem and Ymodem G” on page 185.

Sending Files with Ymodem

To send a file using Ymodem:

- ◇ Begin the Ymodem receive program on the remote computer, specifying a filename if needed. For example, if the remote

computer were a UNIX system running Ymodem, you would enter Terminal mode, begin the remote receive command on the remote computer, and then exit Terminal mode.

- ◇ Select the Filetransfer command from the Online menu and then select the Send command. You will be prompted for the filename. You may enter a single filename from the current directory or a path specification with a single filename; you may use wildcards (see “Wildcards” on page 112).

Receiving Files with Ymodem

To receive a file using Ymodem:

- ◇ Begin the Ymodem send program on the remote computer, specifying a filename if needed. For example, if the remote computer were a UNIX system running Ymodem, you would enter Terminal mode, begin the remote send command on the remote computer, and then exit Terminal mode.
- ◇ Select the Filetransfer command from the Online menu and then select the Get command. The transfer will begin immediately *without* prompting for a local filename.

Zmodem Protocol

BLAST supports the standard Zmodem protocol in both single-file and batch modes. BLAST also supports a variety of special Zmodem features that can be activated through the setup fields of the Zmodem protocol subwindow (page 95).

The following two sections describe interactive file transfers. For a discussion of scripting Zmodem file transfers, see “File Transfers with Zmodem” on page 187.

Sending Files with Zmodem

To send a file using Zmodem:

- ◇ Begin the Zmodem receive program on the remote computer, specifying a filename if needed. For example, if the remote computer were a UNIX system running Zmodem, you would enter Terminal mode, begin the remote receive command on the remote computer, and then exit Terminal mode.

- ◇ Select the Filetransfer command from the Online menu and then select the Send command. You will be prompted for the filename. You may enter a single filename from the current directory or a path specification with a single filename; you may use wildcards (see “Wildcards” on page 112).

Receiving Files with Zmodem

To receive a file using Zmodem:

- ◇ Begin the Zmodem send program on the remote computer, specifying a filename if needed. For example, if the remote computer were a UNIX system running Zmodem, you could enter Terminal mode, begin the remote send command on the remote computer, and then exit Terminal mode.
- ◇ Select the Filetransfer command from the Online menu and then select the Get command. The transfer will begin immediately *without* prompting for a local filename.

NOTE: If the Auto Receive setup field (@ZMAUTODOWN) is set to YES, you do not have to select the Get command; Zmodem transfers the file automatically when you enter Filetransfer mode.

Filename Restrictions

With Xmodem, Ymodem, and Zmodem, you should *not* use any file transfer switches. Also you should not give a file the same name as a switch if forward slashes are used in the pathname—BLAST will either ignore the file or look for a file with the name of the folder containing the file. In either case, the transfer of the file will not occur and you will get an error message. Some of the more common switches to avoid in naming files (uppercase or lowercase) are: app, comp=*n*, follow=*nn*, fwd, group, ovw, owner=*nn*, perms=*nnnn*, str, and txt (where *n* is a number from 0 to 9).

You may work around this restriction by using backslashes if both the local and remote systems support them. If not, you can change your local and remote current directories to the one containing the file you want to transfer and give the filename without a path. Change your local working directory by accessing the Local menu and choosing the Chdir command. Change your remote working directory by accessing Terminal mode and giving the “change current directory” command for that operating system.

Alternatively, you may do a scripting workaround. To change the local working directory use the LCHDIR command. To change the remote directory TSEND a “change working directory” command for that operating system. For example, the following script fragment will change the current remote directory on a UNIX machine to /u/sales.

```
TSEND "cd /u/sales", LF
```

Chapter 10

Text Transfers

Introduction

In BLAST session protocol, you may transfer text directly to and from a remote computer using the respective Online commands Upload and Capture.

Uploading Text to a Remote Computer

Uploading is the process of sending text from your PC to a remote computer. When you upload, the text being uploaded will display on your screen. The receiving computer does not need to be running BLAST, but it must have a program capable of capturing text and responding to flow control.

Because there is no error detection, characters may be dropped or noise may change the characters in the data stream. The following setup fields, however, can assist in regulating the flow of data during text uploads to help prevent the receiving computer from losing characters: Wait for Echo, Prompt Char, Char Delay, and Line Delay. See Chapter 5 for details on using these functions.

After you have connected, there are three ways to start the upload process with another system:

Manual Method

- ◇ Select Terminal from the Online menu.
- ◇ Type the appropriate commands for the remote computer to start a text capture program. Note that an entry is not required in the System Type setup field for this method.
- ◇ When the remote capture program is ready, press *ATTN ATTN* to exit Terminal mode and then select Upload from the Online menu. Specify the desired local filename, *but not a remote filename*.
- ◇ After the upload is completed, you will be returned to Terminal mode. Save the file containing the newly captured text, specifying a name if you have not already done so on the command line, and then quit the capture program.

Interactive Automatic Method

Select the Upload command from the Online menu. *You must specify both the local and remote filenames.* Your PC will automatically send the file to the remote system if text capture is supported by that system.

NOTE: The remote computer type must be entered in the System Type setup field for this method to work because BLAST uses the SYSTEMS.SCR library to automate the process. BLAST will start the remote text capture program for you.

BLASTscript Automatic Method

See “Text Transfers” on page 190 for details on scripting uploads.

Downloading Text from a Remote Computer

Downloading is the process of capturing text sent from another system to your computer. When you capture text from a remote computer, the text being downloaded will display on your screen. The sending computer does not need to be running BLAST, but it must have a program capable of sending text and responding to flow control. If flow control is specified in the setup, BLAST will pause

transmission for a few moments when the buffers are full. After connecting, there are two ways to start the download process:

Manual Method

- ◇ Select the Capture command from the Online menu and specify the desired filename for the capture file.
- ◇ Select Terminal from the Online menu. Type the appropriate command for the remote computer to start typing the text. For example, at the “\$” prompt on a UNIX system, you might type:

```
cat test.fil
```

- ◇ When the download has completed, press *ATTN ATTN* to exit Terminal mode. Turn Capture off by selecting it again.

BLASTscript Automatic Method

See “Text Transfers” on page 190 for details on scripting downloads.

Chapter 11

BLAST Editor

Using BLAST Editor

BLAST editor is available through the Edit command of the Local menu. Using it, you can modify BLAST scripts or any ASCII text file. Files created using BLAST editor contain no special formatting codes, such as overstrike, underline, bold, or word wrapping. The maximum file size that can be edited with BLAST editor is 57996 bytes.

If a file is changed, the original file is renamed with a “.BAK” extension. You may recover the previously saved version of the file by exiting BLAST editor, deleting the current version of the file, and renaming the “.BAK” file the original name.

If you prefer a different text editor, you may link your editor to the Edit menu command by setting the EDITOR option in BLAST.OPT (see page 24).

NOTE: Some systems use the CTRL S and CTRL Q characters for flow control. Also note that, in a LAN environment, when you specify a filename to edit with BLAST Editor, you should always prefix the filename with a full path. If the file is in the current directory, use

the prefix “.” (period backslash) to insure that the file is saved to the correct directory.

Quick Reference

This section provides a quick reference to BLAST Editor key sequences. Note that quick movement keys included in the list below require pressing CTRL Q and then a letter. Similarly, blocking keys require pressing CTRL K and then a letter.

Cursor Movement

CTRL S	or	←	cursor moves space to the left
CTRL D	or	→	cursor moves space to the right
CTRL A			cursor moves word to the left
CTRL F			cursor moves word to the right
CTRL I			cursor moves to next tab stop (insert mode off)
CTRL Q S	or	F5	cursor moves to start of line
CTRL Q D	or	F6	cursor moves to end of line
CTRL E	or	↑	cursor moves to preceding line
CTRL X	or	↓	cursor moves to next line
CTRL Q G	or	F8	cursor moves to line number
CTRL Q E	or	HOME	cursor moves to top of screen
CTRL Q X	or	END	cursor moves to bottom of screen
CTRL R	or	PG UP	cursor moves up a page
CTRL C	or	PG DN	cursor moves down a page
CTRL Q R	or	F9	cursor moves to start of file
CTRL Q C	or	F10	cursor moves to end of file
CTRL Q P			cursor moves to previous position

Scrolling

CTRL C			scroll down one screen
CTRL R			scroll up one screen
CTRL W			scroll up one line
CTRL Z			scroll down one line

Insertion/Deletion

CTRL V			toggle insert mode on/off
CTRL P			insert control character
CTRL N			insert blank line
CTRL K R			insert contents of file at cursor position (you will be prompted for a filename)
CTRL G	or	DELETE	delete character
CTRL H			delete preceding character
CTRL T			delete word

CTRL Y	delete line
CTRL Q Y	delete text from cursor to end of line
CTRL K Y	delete block
CTRL I	move text on right to next tab stop (insert mode on)

Blocking

CTRL K B or F3	mark beginning of block
CTRL K K or F4	mark end of block
CTRL K H or F2	unmark block
CTRL K C	copy block
CTRL K Y	delete block
CTRL K V	move block to current cursor position
CTRL K P	print block
CTRL K W	write block into a file (you will be prompted for a filename)

Searching

CTRL Q F or F7	find word or phrase
CTRL Q A	find and replace word or phrase
CTRL L	find and/or replace word or phrase again

Quitting

CTRL K X	save file and exit
----------	--------------------

Cursor Movement and Scrolling

The cursor in BLAST Editor can be positioned to any location within the file. You may not, however, place the cursor in locations that do not contain characters. You may view text without moving the cursor by using several scrolling keys. For a list of key sequences for cursor movement and scrolling, see “Quick Reference” in the preceding section.

Inserting and Deleting Text

BLAST Editor enters text in two ways. When insert mode is on, added characters move all other characters to the right, thus inserting the text. When insert mode is off, typed characters replace existing characters. CTRL V toggles insert mode on and off. When insert mode is on, “ins” appears at the top of your screen. BLAST editor begins with insert mode on. For a list of key sequences for inserting and deleting text, see “Quick Reference” above.

Managing Text Blocks

BLAST Editor block keys provide an easy-to-use cut and paste capability. A “block” is a specified area of text that can be modified in a single operation.

To define a block, you must first mark the beginning and the end of the text to be blocked. Use `CTRL K B` to mark the beginning of the block and `CTRL K K` to mark the end of the block. The marked text will be highlighted on the screen. Once the text is defined, you may copy, delete, move, write, or read the block of text. If you make a mistake while deleting text, you can recover the lost text by retyping or by copying the .BAK file. Below are key sequences for editing blocks of text:

<code>CTRL K B</code>	or	<code>F3</code>	mark beginning of block
<code>CTRL K K</code>	or	<code>F4</code>	mark end of block
<code>CTRL K H</code>	or	<code>F2</code>	unmark block
<code>CTRL K C</code>			copy block
<code>CTRL K Y</code>			delete block
<code>CTRL K V</code>			move block to current cursor position
<code>CTRL K P</code>			print block
<code>CTRL K W</code>			write block into a file (you will be prompted for a filename—if a file with that name already exists, you will be asked if you want to overwrite it; if a file with that name does not exist, it will be created)

Searching

Searching text provides a convenient way to move the cursor to a desired place in a file without inspecting the entire file on the screen. Searching and replacing text provides an easy way to find and replace one word, phrase, or character string without explicitly positioning the cursor, deleting the old text and typing in the new. The following key sequences are used in searching:

<code>CTRL Q F</code>	or	<code>F7</code>	find word or phrase
<code>CTRL Q A</code>			find and replace word or phrase
<code>CTRL L</code>			find and/or replace word or phrase again

Finding Text

To find a particular string of text, press the Find key, CTRL Q F or F7. You will be prompted for the search string. After entering the search string, you will be presented with the following search options:

<i>a=all</i>	find/replace all occurrences
<i>b=backward</i>	search backward from cursor position
<i>k=within marked block</i>	search within marked block
<i>u=ignore case</i>	make search case-insensitive

Enter a, b, k, or u or any combination of these to specify your search option(s). BLAST Editor will then find the first occurrence of the search string, placing the cursor at the end of the string (unless you choose the “a” option). After making any edits, simply press CTRL L to find the next occurrence of the text string.

Finding and Replacing Text

To find and replace text, press CTRL Q R. You will be prompted for the search string and replacement text. After entering these, you will be presented with the same options as noted above for find. After you have specified any options, BLAST Editor will find and highlight the first occurrence of the text string (unless you choose the “a” option). You will then be presented with the following options:

^Y replace & find next *^N find next* *^A find and replace all*

If you make any edits instead of choosing one of these options, you may resume your search-and-replace without re-entering the search and replace strings by using CTRL L key sequence.

Quitting BLAST Editor

When you have finished entering text, press ESC. At the prompt, press one of the following: Y to save the file and return to the Local menu, N to exit to the Local menu without saving, or ESC to resume editing. Alternatively, after entering text, you may press CTRL K X to save the file and exit.

Chapter 12

Introduction To Scripting

Starting Out

Scripts allow BLAST to automate communications tasks. Scripts are often used for tasks such as logging into remote hosts and handling the details of communications sessions that are repetitive or that inexperienced users would find overwhelming. This chapter introduces the BLASTscript language and describes an important feature of BLAST that aids scripting—Learn Mode. With Learn mode, BLAST writes your scripts so that learning scripting is made easier.

Executing BLAST Scripts

BLAST scripts can be invoked using one of three different methods.

- ◇ From the Online menu, select the Script command. When prompted for the script name, enter the name of the file. This interactive method of starting a script is preferable when you wish to automate only a portion of your communications session.
- ◇ In a setup, enter the name of a BLAST script in the Script File field. After the setup is loaded into memory and the Online command is selected from the Offline menu, the script named

in the setup will execute automatically. This is useful if you always use a specific script with a particular setup.

- ◇ From the operating system command line, specify a BLAST script name with the `/s` switch (see “Command Line Switches” on page 10). The script specified on the command line takes precedence over a script listed in the Script File setup field.

You can include a directory path when you specify a script filename. If you do not name a directory, BLAST will first search the current directory and then the `SETUPDIR` directory.

To abort a script completely, press `ATTN ATTN`. To abort a script after the currently executing statement completes execution, press `ATTN` once.

Writing a Script

The best way to learn how to write a script is by doing it. First, start BLAST Editor, another text-file editor on your computer, or a word processing program. If you prefer to use a word processor for creating script files, be aware that your scripts must always be saved as text files, not word processor documents. Your scripts should be saved in the directory from which you normally operate BLAST, or in the BLAST directory. These are the only two locations where BLAST will search for script files unless you specify a search path.

If you choose to use BLAST’s own text-file editor for writing your scripts, select the Edit option from the Local menu and enter a valid filename, such as:

```
HELLO . SCR
```

The `.SCR` extension identifies the file as a script file. The extension is not required, but it is recommended.

You can now enter text as you would in a word processor. For a summary of the editing commands available in BLAST’s editor, see Chapter 11.

After starting the editor, type in the following short script:

```
# HELLO.SCR
#
# Just wanted to say hi
#
.begin
```



```
    display "Hello, world!"
    return
#
# End of script.
```

Save this file and go to BLAST's Online menu. Choose the Script option and enter the filename

HELLO

When HELLO.SCR executes, it displays the message

Hello, world!

on your screen and then returns control to you.

About HELLO.SCR

As simple as HELLO.SCR is, it illustrates several important scripting concepts. All the lines starting with “#” are comments explaining the functions of the script commands and are not displayed. You may be surprised how quickly you can forget why you wrote a particular script or how an especially difficult section of code actually works. Comments can clarify what you are trying to accomplish with your script.

In HELLO.SCR, the line beginning with a period, `.begin`, is called a label. A label serves not only as a supplemental comment, but also as the destination for a `GOTO` command, discussed later.

The `DISPLAY` command causes text to be displayed on your computer screen; it does not cause text to be transmitted through the serial port. Another script command, introduced later, performs this task.

Finally, the `RETURN` command returns control of BLAST to you.

A Sample Script

To learn more about scripting, it is helpful to imagine a problem that can be solved through scripting. Suppose an art supply company has several retail stores and a central headquarters. Each evening, the stores must send an updated inventory to the central office by modem.

Pam, the system administrator for the central office VAX computer, determines that the local managers should dial into the VAX at their

convenience and use BLAST protocol to transfer their files. She has assigned a unique filename on the VAX for each store's inventory information—STORE01.DAT, STORE02.DAT, and so forth. Each store keeps its inventory in a file called INVNTRY.DAT. For each local manager, Pam provides a script to do the following:

1. Connect to the remote VAX system
2. Send the inventory file to the VAX
3. Disconnect

A script that meets these requirements is illustrated below. The script DAILYRPT.SCR is certainly more complicated than HELLO.SCR, but the same three sections that originally outlined are present. To make it easier to discuss the script, we will refer to the line numbers shown in brackets next to the script statements. You would *not* include these numbers in *an actual* script.

CONNECTING (Section 1)

The first section of the script (`.begin`) establishes the connection with the head office. Line 9 sets a variable, called `@ONERROR`. In a BLAST script, all variables begin with “@”. Some variables are reserved variables, meaning that they are defined by BLAST for special purposes; other variables can be created by you (see Chapter 16). `@ONERROR` is a reserved variable that determines how BLAST will respond to routine (nonfatal) errors. By giving `@ONERROR` the value `CONTINUE`, Pam is telling BLAST to skip error messages rather than pause and wait for a human operator to respond.

```
[ 1] # DAILYRPT.SCR
[ 2] #
[ 3] # A script to send daily inventory reports to the
[ 4] # head office
[ 5] #
[ 6] # Section 1: CONNECTING
[ 7] #
[ 8] .begin
[ 9]   set @ONERROR = "CONTINUE"
[10]   connect
[11]   if @STATUS = "0" goto .xfer
[12]   display "No Connection! Error code: ", @STATUS
[13]   return
[14] #
[15] # Section 2: TRANSFERRING
[16] #
[17] .xfer
```

```

[18] filetransfer          # enter BLAST protocol
[19] send                  # prepare to send a file
[20] invntry.dat          # local filename
[21] store02.dat          # remote filename
[22] t                    # specify text file
[23] esc                  # exit Filetransfer mode
[24] if @EFERROR not = "0"
[25]     display "An error occurred during file transfer."
[26]     display "Please examine the log file."
[27] end
[28] #
[29] # Section 3: DISCONNECTING
[30] #
[31] .finish
[32] disconnect
[33] return
[34] #
[35] # End of script.

```

Line 10, the `CONNECT` statement, is responsible for a great deal of work. The `CONNECT` statement, like Connect from BLAST's interactive menus, initializes the modem, dials the head office, and enters the username and password for the store's VAX account. All of this information—the modem type, phone number, remote system type, and account information—is taken from the setup (see *Connecting and Disconnecting* on page 193).

Line 11 demonstrates how scripts are programmed to make choices with the `IF` (conditional) statement. After the `CONNECT` command executes, it sets the value of `@STATUS` to indicate whether or not the connection was successful. The `IF` statement tests the value of `@STATUS` in its conditional clause. If `@STATUS` equals 0, the connection was successful and the script performs the `GOTO` command, sending the script to the section labeled `.xfer`, which controls file transfer.

```
if @STATUS = "0" goto .xfer
```

<i>conditional statement</i>	<i>executes if conditional clause is true</i>
----------------------------------	---

If `@STATUS` equals any value other than 0, script execution continues on line 12, displaying “No Connection” and an error code. At this point, `RETURN` aborts further execution of the script and control is returned to the user.

TRANSFERRING (Section 2)

The second section, under the `.xfer` label, begins with the `FILETRANSFER` statement. The `FILETRANSFER` statement works like the Filetransfer command of the Online menu. When it is executed, BLAST attempts to start the BLAST software on the remote computer, and the script pauses until Filetransfer mode is entered or a time limit expires. The exact events that occur when `FILETRANSFER` command is executed depend on the setting of the System Type setup field (page 66).

The next four lines (19–22) provide the information BLAST protocol needs to send the required file as a text file. If another protocol were used, this section would be scripted differently (for more information on scripting for alternative protocols, see Chapter 13). Line 23, `ESC`, ends the filetransfer session.

Lines 24–27 illustrate another form of the `IF` command, `IF-END`. With `IF-END`, several lines of script can be executed in a block if the conditional clause is true. In line 24, the `@EFERROR` reserved variable is tested, which indicates if any errors occurred during a BLAST protocol file transfer. If `@EFERROR` equals 0, no errors were encountered. For any value other than 0, two messages (lines 25–26) are displayed and the `IF` statement ends. In either case, the script advances to the `.finish` label.

DISCONNECTING (Section 3)

The final section of the script, under the `.finish` label, begins with the `DISCONNECT` command. Like `CONNECT` and `FILETRANSFER`, `DISCONNECT` performs the same operation as the corresponding command of the Online menu. As you become more familiar with BLAST's scripting language, you will discover that many script commands are similar to the options on BLAST's interactive menus. `RETURN` ends the script and returns control of BLAST to you.

Learn Mode

An important aid to writing your own scripts is BLAST's Learn mode. With Learn, you perform a communications task exactly as the script should perform it, and BLAST creates the script from the actions you take. Typically, the Learn script serves as a rough draft of the final script.

To start Learn mode, select Learn from the Offline menu. BLAST will prompt you to name the Learn script. The status indicator

“LRN” will appear in the Status Line at the bottom of the screen to remind you that your keystrokes are being recorded.

Suppose that you wanted to write a script to log into a computer for which there is no standard system type in the BLAST setup. A bank’s computerized account service, for example, may have an unusual login. Assume that after the modems connect, the bank issues the prompt “MIDAS>,” waits for your user identification (AlbertyArtCo), and then issues the prompt “?:”.

To help you write your login script, start Learn mode and then proceed to log in as usual, being careful to avoid spelling errors and other trivial mistakes. When you finish, return to the Offline menu and select Learn again to turn off Learn mode.

An example of what the Learn script might look like is shown below:

```
# BLAST Learn mode script
# Original filename: BANK.SCR
# Date: 01/1/99
# Time: 11:00:00
#
CONNECT
# entering TERMINAL mode
#
ttrap 6, "\012\015MIDAS>"
tsend "Alber"
tsend "tyArtCo" CR
ttrap 3, "\012\015\012\015\?:"
# exiting TERMINAL mode
# RETURN commented out for appending
```

Even though the script has a strange appearance, you can decipher it. TSEND is the script command for transmitting text through the serial port. This command is used for sending the user ID to the bank. TTRAP is used for checking text coming into the serial port, so it is used for detecting the prompts issued by the bank’s system. Without doing any more work, this script will actually perform the login.

Editing the Learn Script

Because BLAST cannot distinguish the meaning of any of the data entering or leaving the serial port, Learn mode may “break” strings of text inappropriately. Editing the Learn script to make the TSEND statements meaningful to human readers is a good idea, but it is not necessary. Likewise, TTRAP statements may contain unneeded

characters when scripted by Learn mode. In the example above, \012 is the octal representation of the line feed and \015 is the octal form of the return character. These characters are not needed to detect the prompts issued by the bank, so they may be edited for clarity.

After you have cleaned up the Learn script, it could look like this:

```
# BANK.SCR
#
# A script to log into the bank
#
.begin
CONNECT
ttrap 6, "MIDAS>"
tsend "AlbertyArtCo", CR
ttrap 3, "?:"
return
#
# End of script.
```

Now the script can be read more easily. After connecting, the script will wait for up to six seconds for the string "MIDAS>." Next, the script will send the string "AlbertyArtCo" and a carriage return. Finally, the script will wait for up to three seconds for the "?:" prompt and then return control to you.

Polishing the Learn Script

After being edited, the Learn script makes better sense to human readers, but it can still be improved. Take a moment to assess it. What's left to be done?

One area for improvement is in error handling. You saw earlier that @STATUS could be tested after the CONNECT command to determine whether a connection was established. Similar error checking should be added to the Learn script.

Another area for improvement is in the use of variables. At present, the user ID is "hard-coded" into the script, meaning that it has a fixed value. If the userid is placed in the appropriate field of the set-up, the script can access it with the @USERID reserved variable. Thus, a more polished version of the Learn script might look like:

```
# BANK.SCR
#
# A script to log into the bank
#
.begin
CONNECT
if @STATUS not = "0" return
ttrap 6, "MIDAS>"
tsend @USERID, CR
ttrap 3, "?:"
return
#
# End of script.
```

As you can see, Learn mode and your own knowledge of BLAST's scripting language simplify the process of automating your communications tasks.

Writing Your Own Scripts

You have now seen enough of the scripting language to begin writing your own scripts. You may wish to read Chapter 13, which describes techniques for working with disk files, manipulating strings, and interacting with programs in your system. Chapter 14 discusses the BLAST method of connecting and disconnecting, which relies heavily on scripts. Chapters 15 and 16 serve as reference guides for all scripting commands and reserved variables, respectively. Many examples are included in these chapters to help you get started. In addition, sample scripts are available for download from Blaster (see "Connecting to Blaster" on page 44).

Chapter 13

BLASTscript Topics

Scripting Basics

Although scripts can address a wide range of communications needs, most scripts handle a limited number of common tasks, such as capturing text to a file, displaying information on the screen, and communicating with other programs in the computer. In this chapter we will demonstrate scripting techniques for such tasks.

Programming Style

It may sound strange to say that a script should conform to a certain “style,” but following a logical style will make it easier for others to understand your script. For example, indenting sections of script that execute together, such as the code in a conditional (`IF-END`) block, is a simple stylistic convention that helps readability, as in the following script:

```

# Start of script
#
.begin
display "Hello, world!"
if @EMULATE = "TTY"
    display "Your emulation is set correctly"
end
else
    set @EMULATE = "TTY"
    display "Your emulation is now TTY"
end
return
#
# End of script

```

Your programming style also affects how efficiently the script will execute. BLAST scripts are interpreted, meaning that BLAST deciphers the instructions in each line of your script as it executes. To make your script run most efficiently, you should:

- ◇ Use spaces between expressions. For instance, the script interpreter can evaluate the first line in the example below more easily than it can the second line because of the spaces placed around “=”.

```

if @STATUS = "0" set @mystat = "GO"

if @STATUS="0" set @mystat="GO"

```

- ◇ If certain labels in your script will be frequent destinations for the GOTO command, place those labels near the beginning of the script. BLAST looks for labels from the start of the script and works down.

Legal and Illegal Expressions

An error that you may encounter during script development is “illegal menu selection.” This error indicates that BLAST has encountered a command in your script that it could not execute. Every line in a script must be executable or contain a comment preceded by #. Blank lines are rarely executable (except for special cases discussed later); thus, do not use blank lines in a script to separate lines of code visually. If BLAST encounters a blank line in a script where it is unexpected, the script interpreter will generate the “illegal menu selection” error.

ILLEGAL

```
if @STATUS = "0"  
  
disconnect  
  
end  
return
```

LEGAL

```
if @STATUS = "0"  
#  
disconnect  
#  
end  
return
```

A typing mistake in a script line can also generate an error message. For example, a line such as

```
ig @STATUS = "0"
```

will generate the “illegal menu selection” error because “ig” is not a valid script command.

The Status of @STATUS

The result of many script operations is reported in the reserved variable @STATUS, which has a number of functions, including indicating whether an error occurred during the CONNECT command and identifying which item in a list of target strings was detected by TTRAP. Because @STATUS is affected by so many script operations, you may need to save the value of @STATUS in a “safe” variable so that you can refer to it later in your script, as in the following example:

```
# Following is the target list:  
#  
ttrap 5, "Apples", "Oranges", "Peaches"  
#  
# Save @STATUS in a user-defined variable.  
#  
set @fruit = @STATUS  
#  
# @STATUS will be changed below by the DISCONNECT statement  
#  
disconnect  
if @STATUS = "0" display "Disconnected OK"  
else display "Disconnect failure!"  
if @fruit = "0" display "No fruit was selected"  
if @fruit = "1" display "Apples are delicious"  
if @fruit = "2" display "Oranges are tasty"  
if @fruit = "3" display "Peaches are nice, too"  
return  
#  
# End
```

For a list of all the commands that set @STATUS, see “Commands That Set @STATUS” on page 204.

The CALL Command

When you set out to write a complicated script, ask yourself whether the script is made up of logically distinct sections. If so, you may be able to code each section as a separate script and write a “master” script that calls each section as required, checking for errors. Working with several small scripts is generally preferable to a single large one because it is easier to follow the logic of the program and find errors. The CALL command is used to transfer execution to another script. For example,

```
call "GETDATA"
```

calls the script named “GETDATA.” When the RETURN command is executed in the called script, control returns to the calling script:

```
return [exit_code]
```

The exit code is optional. When control is returned to the calling script, the value of @STATUS in the calling script is equal to the value of the exit code. For example, the script TESTONE.SCR would call the script TESTTWO.SCR as follows:

```
# TESTONE.SCR
#
#   display "This script calls TESTTWO.SCR"
#   call "TESTTWO.SCR"
#
#       ...
```

At this point, TESTTWO.SCR executes:

```
# TESTTWO.SCR
#
#   ask "Enter a number: ", @input
#   return @input
#
# End
```

The value of @STATUS in TESTONE.SCR has now been set to the value of @input entered in TESTTWO.SCR, and TESTONE.SCR continues with the remainder of its commands:

```

# ...
  display "Now @STATUS = ", @STATUS
  return
#
# End

```

A script that has been called may call another script, a process known as “nesting.” Scripts may be called recursively to the limit of available system resources.

All variables in a script are global, meaning that they can be read and changed anywhere. For example, you can write a script that only sets the variables you will use. Your “master” script then calls this script at the beginning of execution. The master script and any other scripts you call afterward will “see” the variables that you created.

Executing in a Loop

To create a loop, you can write a script to keep track of a loop counter and use the GOTO command:

```

# looping demo number 1
#
  set @count = "10"
.loop
  display "Countdown: ", @count
  let @count = @count - "1"
  if @count not = "0" goto .loop
  display "BLAST off!"
  return

```

Running the script would result in the following display on your screen:

```

Countdown: 10
Countdown: 9
Countdown: 8
Countdown: 7
Countdown: 6
Countdown: 5
Countdown: 4
Countdown: 3
Countdown: 2
Countdown: 1
BLAST off!

```

An alternative method of looping uses the REPS command. With REPS, the previous script could be written as:

```
# looping demo number 2
#
  reps 10
.loop
  display "Counting down..."
  if reps goto .loop
  display "BLAST off!"
  return
```

Since testing the value of REPS in an IF statement automatically decrements it, REPS is a more compact way of executing a loop than a loop counter. In the example above, the GOTO statement is executed while REPS is greater than zero, so that the loop is exited after the message “Counting down...” has been displayed 10 times. As shown in the illustration below, this method of writing the script produces a different display than that of a loop counter. Note that if the number of repetitions is taken from a variable, the countdown occurs, but the variable retains its initial value.

```
Counting down...
Counting down...
Counting down...
Counting down...
Counting down...
Counting down...
Counting down...
Counting down...
Counting down...
Counting down...
Counting down...
BLAST off!
```

Manipulating Text

A number of script commands are available for manipulating text files and text strings. The commands that work with text strings include:

STRCAT *string1*, *string2*, [, ...] – Combine two or more strings to make a single, longer string. The longer string replaces *string1*.

`STRINX string1, string2` – Find the first occurrence of *string2* in *string1*. @STATUS holds the position of the first character in *string1* where a match was found.

`STRLEN string1` – Find the length of a string. @STATUS is set to the value of the length.

`STRTRIM, string1, position1, position2` – Extract a substring of *string1* beginning at *position1* and ending at *position2*. After the substring has been extracted, the value of *string1* is set to the substring.

There are other commands for string manipulation, such as the commands to find the ASCII value of a character, to convert all characters in a string to upper or lower case, and to request interactive string input from the user. These and other commands for string manipulation are discussed in Chapter 15.

The following example illustrates the use of string commands:

```
# String demo - extract first and last name from a string
#
# Set variables
#
  set @name = "Johnson, Alfred"
  set @first = @name
  set @last = @name
#
# Find the comma in the name string
#
  strinx @name, ","
#
# Move to last char of last name and extract last name
#
  let @STATUS = @STATUS - "1"
  strtrim @last, 1, @STATUS
  display "Client's last name: ", @last
#
# Move forward to first char of first name and extract
# everything from there to the end of the string
#
  let @STATUS = @STATUS + "2"
  strtrim @first, @STATUS
  display "Client's first name: ", @first
#
# Rebuild full name by concatenating first and last names
#
```

```

strcat first, " ", @last
display "Client's full name: ", @first
return
#
# End of script.

```

Capturing Text

Two commands, `TCAPTURE` and `SETTRAP`, are available for capturing text as it enters the serial port. The `TCAPTURE` command is used if the text is to be placed in a disk file. The following script illustrates a simple implementation of `TCAPTURE`.

```

# Capture demo
#
#   tcapture on "SALES.RPT"
#
# Pause script until 4 sec of "quiet" elapses
#
#   wait 4 idle
#   tcapture off
#
# End of script.

```

The `TCAPTURE` command itself does not initiate the text capture. Text capture starts when a `WAIT`, `TSEND`, `TTRAP`, or `TUPLOAD` command is executed.

The second method, `SETTRAP`, allows incoming text to be captured into a script variable. The `SETTRAP` command itself does not cause any text to be captured, but it prepares `TTRAP` to capture text by setting a variable into which the captured text is to be saved and specifying a limit on the number of characters saved into the variable. A simple form of `SETTRAP/TTRAP` is:

```

# Settrap/ttrap demo
#
#   settrap @input, 65   # Capture up to 65 char till end of
#                       # line reached
#
#   ttrap 30, "^M^J"
#
# End of script.

```

In this example, up to 65 characters are saved into the variable `@INPUT`. The string `^M^J` (carriage return/line feed) triggers the end of the captured text, which includes the trigger string and any text preceding the trigger—up to 65 characters. If no incoming char-

acters match the trigger within 30 seconds, the last 65 characters of text are saved to the variable @INPUT.

More complex forms of the TCAPTURE and SETTRAP commands are described in Chapter 15.

Reading and Writing Text Files

A script can read and write entire lines of text from a text file. As many files can be open at a time as there are file handles available in your system. The commands for opening a file are:

FOPENA *handle*, *filename* – Open a file for appending.

FOPENR *handle*, *filename* – Open a file for reading.

FOPENW *handle*, *filename* – Open a new file for writing (overwrites existing file).

These commands must specify two pieces of information: the filename and a file handle. The file handle is an integer that other commands in the script will use to refer to the file. @STATUS is set to the value 0 if the file is opened successfully.

The commands for reading, writing, and closing files are:

FREAD *handle*, *variable* – Read a line of text.

FWRITE *handle*, *string* [, *string*] – Write a line of text.

FCLOSE *handle* – Close the file.

To be read properly, a line of text cannot be longer than the maximum length of a variable, which is 139 characters. When read and write operations are successful, @STATUS is set to 0. If they are unsuccessful—for example, a script attempts to read past the end of a file—@STATUS is set to a nonzero value.

Following is an example of a script that uses file handling commands:

```
# File read/write demo
#
# Open MODEMS.SCR and count the number of lines.
# Write the result in a new file called LINE.CNT.
#
```

```

.begin
  clear
  set @file = "MODEMS.SCR"
  fopenr 1, @file
  if @STATUS not = "0"
    werror "Can't open MODEMS.SCR"
    return
  end
  fopenw 2, "LINE.CNT"
  set @count = "0"
  display "One moment, please."
  cursor 10, 6
  put "Reading line #"
.loop
  fread 1, @input
  #
  # If @STATUS is 0, count line and return for another
  #
  if @STATUS = "0"
    let @count = @count + "1"
    cursor 10, 21
    put @count
    goto .loop
  end
.continue          # end of file!
  fwrite 2, @count, " lines in MODEMS.SCR."
  fclose 1
  fclose 2
  display "Done! Check LINE.CNT for line count."
  return
# End of script.

```

Managing the Screen Display

Thoughtful screen displays help users gain a sense of being “in good hands.” Informing users of the progress of a lengthy job, such as a file transfer, frees them to do other things while the software does its job. Displaying too much text onto the screen at once or neglecting the screen completely, however, can make users wonder instead if their session has malfunctioned. BLAST’s scripting language provides a number of commands and reserved variables for controlling the screen to present the right amount of information.

Turning Off the Screen

For some applications, you may wish to turn off regions of the screen while running a script. (To disable screen displays altogether, include the `/n` switch on the command line when you start BLAST; see “Command Line Switches” on page 10) The following reserved variables control particular regions of the display:

`@USERIF` – The user interface area, or menu area, at the top of the screen.

`@SCRLREG` – The scrolling region in the middle of the screen.

`@STATUSLN` – The status line at the bottom the screen.

`@TRANSTAT` – The File Transfer Status Area of the screen.

Set these variables to 0 or OFF to disable the corresponding screen areas. Set the variables to 1 or ON to enable them. For example, if you do not want the BLAST menus to be displayed while your script is running, you would put the statement

```
set @USERIF = "0"
```

in your script. The top four lines of the display would then become part of the scrolling region.

IMPORTANT: You must remember to turn the menu region back on in the script or the user will *not* be able to see the BLAST menus after the script is finished.

Displaying Text in the Menu Region

Two script commands permit you to display text in the menu region:

`WRITE string [, string]` – Prints a message.

`WERROR string [, string]` – Prints a message in the menu region and then waits for the user to press a key. (The script will not pause if `@ONERROR` is set to `CONTINUE`.)

These commands are normally used for displaying errors or progress messages.

Displaying Text in the Scrolling Region

The most common way to display text in the scrolling region is with the `DISPLAY` statement, described on page 209. The `DISPLAY` command prints a string or a list of strings at the current cursor position; depending on the emulation you have chosen, the cursor may or may not advance to the next display line.

Another method of displaying text uses a pair of commands, `CURSOR` and `PUT`:

`CURSOR` *row, column* – Position cursor.

`PUT` *string* [*, string*] – Print string.

The following script demonstrates an application of these commands:

```
# Screen Display Demo
# Hide modem control strings from the user
#
.begin
  set @ONERROR = "CONTINUE"
  set @USERIF = "OFF"
  clear                # Erase the screen
  cursor 12, 30
  put "Now connecting, please wait."
  set @SCRLREG = "OFF"
  connect
  set @SCRLREG = "ON"
  if @STATUS not = "0"
    set @USERIF = "ON"
    clear
    write "Can't connect or log in."
    return
  end
  terminal                # enter Terminal mode
  set @USERIF = "ON"      # don't forget this!
  return
# End of script.
```

Communicating with Other Programs

In some BLAST applications, the end user is not even aware that BLAST is operating in the system. BLAST provides a simple inter-

face that lets other programs control BLAST, hiding the existence of BLAST completely from the user if necessary.

Passing Information to BLAST

The command line can contain up to ten “arguments,” or parameters, that pass information to a BLAST script. Command line arguments follow the setup name on the command line (see “Command Line Switches” on page 10). For example, consider the following BLAST command line:

```
blast chicago /ssales 12:05 midwest
```

This command line will start BLAST with the CHICAGO.SU setup, execute the script called SALES.SCR using the /s switch, and store the arguments “12:05” and “midwest” in the reserved variables @ARG0 and @ARG1, respectively.

A program can also pass information to a script by writing a text file that the script opens and interprets. Alternatively, because a script itself is just a text file, your controlling software can write a script that can be executed by BLAST “on the fly.”

Controlling Other Programs from BLAST

While a script is executing, it can start other programs in your computer with the LOCAL/SYSTEM command (memory permitting). This command allows your script to execute a single command as you would type it on the DOS command line. The following script demonstrates use of the LOCAL/SYSTEM command:

```
# Local System demo
#
# Set the system clock from a script using the
# "Time" system command
#
  set @syscmd = "TIME"
  ask "What time", @input
  strcat @syscmd, " ", @input      # "TIME 2:30"
  local
  system
  @syscmd
  esc
  return
# End of script.
```

File Transfers with BLAST Session Protocol

Chapter 6 describes the BLAST session protocol, including some information about scripting file transfers. This section provides detailed information about writing file transfer scripts.

The coding that performs a file transfer in a script closely follows the sequence of menu choices and prompts that BLAST uses when the same task is performed manually. Thus, it makes sense to practice a communications task interactively before attempting to write the script that will automate the task. Learn mode (page 160) provides another means of getting an idea about how a particular task can be coded in a script.

Getting and Sending Files

A simple GET and SEND could be coded like this (remember, you would *not* include the numbers in brackets):

```
[ 1] filetransfer
[ 2] get
[ 3] yourfile.rpt
[ 4] myfile.rpt
[ 5] ta
[ 6] send
[ 7] labdata.dat
[ 8]
[ 9]
[10] esc
```

in this script, `yourfile.rpt` (line 3) is the response to the Remote Filename prompt that BLAST issues when the `get` command is given, and `myfile.rpt` (line 4) is the response to the Local Filename prompt. The transfer options `t` and `a` (line 5) specify “text” and “append” in this example—the same symbols you would use if you were performing the file transfer interactively.

In the `SEND` example, two blank lines (lines 8 and 9) are entered to specify that BLAST use default values for these responses. Thus, the remote filename will be the same as the local filename, and no transfer options are specified (the file transfer will be binary). *Blank lines representing default filenames and file attributes (t, o, a) cannot contain comments.*

Other than the preceding exceptions, you should not have blank lines in a script unless they contain the comment character, `#`. The

ESC statement represents pressing the ESC key, which is the action that you normally take to exit Filetransfer mode.

Performing Remote Commands

The BLAST session protocol allows you to perform remote system commands without special knowledge of the command syntax of the remote machine. Remote commands are coded in a script like this:

```
filetransfer
remote
  chdir
  /usr/customer
  esc
esc
```

The first ESC represents the ESC keystroke that will move you from the Remote menu to the Filetransfer menu. The second ESC terminates the session in the usual manner.

Using Transfer Command Files

A powerful feature of the BLAST session protocol is the ability to take its commands from a transfer command file (see “Transfer Command File” on page 117). To use a transfer command file in a script, the following syntax is used:

```
filetransfer
file
transfer.tcf
esc
```

where *transfer.tcf* is the command filename. The extension .TCF is often used to identify a transfer command file, but this convention is not required.

Sending Messages

BLAST protocol can send messages between systems during a BLAST session (see the description of the Message menu option on page 110). String-variables may be substituted for messages.

```
filetransfer                               # issue the transfer command
message                                   # sending a message
  Sending Sales Reports                     # the message
esc                                         # exit Filetransfer mode
```

Special Considerations

To take full advantage of the BLAST session protocol, keep the following points in mind:

- ◇ BLAST attempts to queue as many remote commands as possible (like GETs) before issuing local commands (like SENDs). This behavior permits BLAST to transmit files in both directions simultaneously, but it also means that files may not be transmitted in the order specified in the script.
- ◇ Many filetransfer and file management commands can be combined into one FILETRANSFER-ESC block, as in the following example:

```
filetransfer      # begin Filetransfer mode
send              # send files that
*.TXT            # match the template
%
ta
remote           # begin remote file mgmt
  chdir
  /usr/customer
  print
  client.log
  esc            # leave remote file mgmt
file             # use a command file
site3.tcf
esc              # exit Filetransfer mode
```

Combining operations allows BLAST to work more efficiently, saving online charges or other long-distance telephone costs.

- ◇ Errors that occur during file transfer can be checked by testing the value of @EFERROR or by examining an @EFLOG file after exiting Filetransfer mode. If Extended Logging is enabled, additional reserved variables give information about the number of successful transfers and the number of failures. These reserved variables are described in Chapter 15. See also “Using Log Files for Error Checking” on page 188.

If the line is dropped during a file transfer, BLAST can either ignore the problem or abort Filetransfer mode immediately. The action BLAST takes is determined by the setting of the DCD Loss Response setup field. Your choice in setting DCD Loss Response (ABORT or IGNORE) may depend on your environment. For instance, you might need to ignore carrier loss if the

carrier drops intermittently due to line noise and your modems are configured to retrain automatically.

File Transfers with FTP

The syntax for FTP file transfers is the same as for BLAST protocol except that there are no transfer options; therefore, there is no line for transfer options in FTP scripts. The basic file transfer syntax is:

```
filetransfer
send
local_filename
remote_filename
get
remote_filename
local_filename
esc
```

As with BLAST protocol, a blank line for the receiving filename indicates that the file will retain its original name. For example, in the following script

```
filetransfer
get
newinventory.txt

esc
```

the local filename will remain the same as the remote filename—`newinventory.txt`.

File transfer scripts can be improved by adding error-checking features. For a discussion of error checking in file transfer scripts, see “Using Log Files for Error Checking” on page 188.

File Transfers with Kermit

Before writing scripts for Kermit, you may want to review the general information in Chapter 8, *Kermit Protocol*, on page 131. Learn mode (page 160) is also a good tool for obtaining a rough draft of a script you will need in a particular case. The scripting syntax shown below for Kermit assumes a connection to a multi-user system such as UNIX.

Sending Files

Before issuing a SEND command, you must start simple Kermit or Kermit server on the remote machine.

Simple Kermit

After starting simple Kermit, you must issue a SEND command on the remote machine. The basic syntax for sending files using simple Kermit is as follows (the actual receive command depends on the specific implementation of simple Kermit):

```
connect
tsend "kermit", LF
tsend "receive_command local_filename", LF
filetransfer
send
local_filename
esc
```

Kermit Server

Before issuing a SEND command, you must start Kermit server on the remote machine. For most UNIX machines, this command is "kermit -x." The basic syntax for sending files using Kermit server is as follows:

```
connect
tsend "kermit -x", LF
filetransfer
send
local_filename
remote_filename
esc
```

Receiving Files

Kermit has been implemented on many computer systems. BLAST's implementation of Kermit supports both "receiving" and "getting" files from remote computers. The RECEIVE command is used to transfer a file from simple Kermit, whereas a GET command is used for transferring a file from a Kermit server.

Simple Kermit

Before issuing a RECEIVE command, you must start simple Kermit on the remote machine and issue a send command. The basic syntax for receiving files using simple Kermit is as follows (the actual send command depends on the specific implementation of simple Kermit):

```
connect
tsend "kermit", LF
tsend "send_command remote_filename", LF
filetransfer
receive
local_filename
esc
```

Kermit Server

Before issuing a GET command, you must start Kermit server on the remote machine. For most UNIX machines, this command is "kermit -x." The basic syntax for GETs using Kermit server is as follows:

```
connect
tsend "kermit -x", LF
filetransfer
get
local_filename
remote_filename
esc
```

Transferring More Than One File

Unless you exit simple Kermit or Kermit server on the remote computer, you do not have to issue the command to start Kermit for every transfer block, only the first one. For example, using Kermit server, you could run the following script:

```
connect
tsend "kermit -x", LF
get
salesreport.txt
storesales.txt
send
storelinventory.txt
inventory.txt
esc
tsend "quit", LF
```

For simple Kermit, however, you have to issue the simple Kermit send or get command each time you transfer a file, as in the following example:

```
connect
tsend "kermit", LF
tsend "send salesreport.txt", LF
filetransfer
receive
storelsales.txt
esc
tsend "receive storelinventory.txt", LF
filetransfer
send
inventory.txt
esc
tsend "quit", LF
```

File transfer scripts can be improved by adding error-checking features. For a discussion of error checking in file transfer scripts, see “Using Log Files for Error Checking” on page 188.

File Transfers with Xmodem and Xmodem1K

Before writing scripts for Xmodem and Xmodem1K, you may want to review the general information in Chapter 9 on the use of these protocols. Learn mode (page 160) is also a good tool for obtaining a rough draft of the script you will need in a particular case. The scripting syntax shown below for Xmodem assumes a connection to a multi-user system such as UNIX.

Sending Files

Before issuing a SEND command, you must issue the Xmodem receive command on the remote computer for the remote system’s implementation of Xmodem. The basic syntax for sending a file using Xmodem is:

```
connect
tsend "receive_command remote_filename", LF
filetransfer
send
local_filename
esc
```

Receiving Files

The syntax for receiving files is:

```
connect
tsend "send_command remote_filename", LF
filetransfer
get
local_filename
esc
```

Transferring More Than One File

A separate FILETRANSFER-ESC block is required for each file that is transferred. For example, to send two files and get one file, three FILETRANSFER-ESC blocks are needed, as in the following example:

```
# 3-File Xmodem Transfer
connect
tsend "rx sales.txt", LF
filetransfer
send
slsales.txt
esc
tsend "rx order.txt", LF
filetransfer
send
slorder.txt
esc
tsend "sx inventory.txt", LF
filetransfer
get
slinventory.txt
esc
```

File transfer scripts can be improved by adding error-checking features. For a discussion of error checking in file transfer scripts, see “Using Log Files for Error Checking” on page 188.

File Transfers with Ymodem and Ymodem G

Before writing scripts for Ymodem and Ymodem G, you may want to review the general information in Chapter 9 on the use of these protocols. Learn mode (page 160) is also a good tool for obtaining a rough draft of the script you will need in a particular case. Because the filename is passed to the receiving computer, a filename is not needed when receiving a file. The scripting syntax shown for Ymodem assumes a connection to a multi-user system such as UNIX.

Sending Files

Before issuing a SEND command, you must issue the Ymodem receive command on the remote computer for the remote system's implementation of Ymodem. The basic syntax for sending a file using Ymodem is:

```
connect
tsend "receive_command", LF
filetransfer
send
local_filename
esc
```

Receiving Files

The syntax for receiving files is:

```
connect
tsend "send_command remote_filename", LF
filetransfer
get
esc
```

Transferring More Than One File

A separate FILETRANSFER-ESC block is required for each file that is transferred. For example, to send two files and get one file, three FILETRANSFER-ESC blocks are needed, as in the following example:

```
# 3-File Ymodem Transfer
connect
tsend "rb", LF
filetransfer
send
sales.txt
esc
tsend "rb", LF
filetransfer
send
order.txt
esc
tsend "sb inventory.txt", LF
filetransfer
get
esc
```

File transfer scripts can be improved by adding error-checking features. For a discussion of error checking in file transfer scripts, see “Using Log Files for Error Checking” on page 188.

File Transfers with Zmodem

Before writing scripts for Zmodem, you may want to review the general information in Chapter 9. Learn mode (page 160) is also a good tool for obtaining a rough draft of a script.

The Zmodem protocol is configured through the Zmodem setup sub-window. An important parameter for scripting purposes is Auto Receive. With Auto Receive set to YES in the setup file or the reserved variable @ZMAUTODOWN set to YES in a script, Zmodem will only receive files. Note that a setting for @ZMAUTODOWN in a script overrides the setting of Auto Receive in the setup file.

Because the filename is passed to the receiving computer, a filename is not needed when receiving a file.

The scripting syntax shown for Zmodem assumes a connection to a multi-user system such as UNIX.

Sending Files

Before issuing a SEND command, you must issue the Zmodem receive command on the remote computer for the remote system’s implementation of Zmodem. In the basic syntax for sending a file using Zmodem below, @ZMAUTODOWN, the reserved variable for Auto Receive, is set to NO in case the Setup file has Auto Receive set to YES or @ZMAUTODOWN has been set to YES earlier in the session:

```
set @ZMAUTODOWN = "NO"  
connect  
tsend "receive_command", LF  
filetransfer  
send  
local_filename  
esc
```

Receiving Files

The syntax for receiving files depends on the how you set @ZMAUTODOWN. If @ZMAUTODOWN is set to NO, you need a GET statement:

```
set @ZMAUTODOWN = "NO"
connect
tsend "send_command remote_filename", LF
filetransfer
get
esc
```

If @ZMAUTODOWN is set to YES, you do *not* need a GET statement

```
set @ZMAUTODOWN = "YES"
connect
tsend "send_command remote_filename", LF
filetransfer
esc
```

Transferring More Than One File

As with Xmodem and Ymodem protocols, with Zmodem protocol each FILETRANSFER-ESC block can specify only one file, as in the following example:

```
set @ZMAUTODOWN = "NO"
connect
tsend "rz", LF
filetransfer
send
sales.txt
esc
tsend "sz inventory.txt", LF
filetransfer
get
esc
```

File transfer scripts can be improved by adding error-checking features. For a discussion of error checking in file transfer scripts, see the next section.

Using Log Files for Error Checking

Checking for errors after a file transfer is an important part of a good script. Messages generated during a file transfer are written to the session log file, which you can open and read as you would any other file. For example, the following script automates a BLAST session and checks for errors:


```

set @ftlog = "session.log"
if exist @ftlog ldelete @ftlog
set @LOGFILE = @ftlog
filetransfer
send
orange.txt
fruit.txt
to
esc
set @xferok = "NO"           # initialize user flag
set @LOGFILE = ""          # close session log
fopenr 1, @ftlog           # now open it for reading
.check
fread 1, @logline
if @STATUS = "0"           # successful read
    strinx @logline, "send complete" # crucial!
    if @STATUS = "0" goto .check    # no match
    set @xferok = "YES"           # matched, set user flag
end
fclose 1
if @xferok = "YES" display "Transfer successful"
else display "Could not transfer the file"
return                      # or whatever else

```

Another log file, the error-free log, is available for similar error checking. The error-free log, or “eflog,” contains just the status messages generated during a file transfer and is overwritten each time a FILETRANSFER-ESC block is executed, unlike the session log, which is always appended. Consequently, an eflog can be scanned more quickly than a session log because there are fewer lines to read and discard (see @EFLOG on page 247).

The following script fragment demonstrates how @EFLOG may be used to check for errors.

```

set @EFLOG = "xmodem.log"
filetransfer
get
portland.dat
esc
fopenr 1, @EFLOG # check the log
fread 1, @input  # only 1 line to look at!
fclose 1
strinx @input, "ERROR"
if @STATUS = "0" display "No error occurred."
else display "Error!"

```

The following section describes scripting for text transfers to and from a multi-user system such as UNIX. The receiving computer does not need to be running BLAST, but it must have a program capable of capturing text and responding to flow control. See *Text Transfers* on page 145 for more information about text transfers.

Uploading Text

To upload a text file from within a script, write a BLAST script that includes:

- ◇ a TSEND command to start an editor to capture the data on the remote system and any commands needed for overwriting or appending the file.
- ◇ a TUPLOAD statement (this will honor the setup fields for flow control—XON/XOFF, Wait for Echo, Line Delay, Character Delay, Prompt Character—and linefeed handling). The TUPLOAD command sets @STATUS to 0 if successful; it returns some file I/O errors.
- ◇ a TSEND command to exit the editor on the remote system.

When uploading to a remote computer, remember that some of the data may be buffered. This means that the upload may complete well before all the characters have passed completely to the remote system. Any activity immediately following a TUPLOAD may have to deal with both the trailing characters of the uploaded file and the delay before other activity can be initiated. To avoid these problems, you can:

- ◇ TTRAP for the characters issued by the remote system upon exiting the text editor.
- ◇ Use a WAIT IDLE statement to be sure the buffers have a chance to clear.

The sample script below assumes that the remote computer is running UNIX using the text editor vi. The script TTRAPS for the filename in quotation marks used in vi's exit status line; the WAIT command gives the buffers on the local and remote computers time to clear.

```

connect
tsend "vi cih4", LF # Send cmd to start editor on remote
wait 3
tsend "G", LF      # Move cursor to end of file
tsend "o", LF      # Start new line for appending
tupload "cih4"
wait 3 idle
tsend "\033", LF   # Send escape cmd to remote system
wait 1
tsend ":x", LF     # Send cmd to exit editor on remote
ttrap 30, "\042cih4\042"
set @hold = @status
wait 3 idle
if @hold = "0"
    display "Tupload not completed."
    return
end
else display "Tupload successful."
wait 10

```

For more specific error checking, you can check @STATUS for TUPLOAD:

```

connect
tsend "vi cih4", LF
wait 3
tsend "G", LF
tsend "o", LF
tupload "cih4"
set @hold1 = @status
wait 3 idle
if @hold1 = "0" display "Tupload cmd execution complete."
else
    display "Tupload cmd failure; error ", @hold1
    tsend "\033", LF
    tsend ":q!", LF # Quit editor without saving file
    return
end
tsend "\033", LF
wait 1
tsend ":x", LF
ttrap 30, "\042cih4\042"
set @hold2 = @status
wait 3 idle
if @hold2 = "0"
    display "Tupload not completed."
    return

```

```
end
else display "Tupload completed."
wait 5
```

Downloading Text

To download a text file from within a script, write a BLAST script that includes a TCAPTURE statement. TCAPTURE will receive the specified file from the remote system and activate capture to receive it.

While TTRAP handles a small number of characters for processing by a BLAST script, TCAPTURE accepts large amounts of data and saves it to a disk file. The APPEND option writes the captured data to the end of an existing file or creates a new file. The OVERWRITE option deletes and recreates an existing file or creates a new file. If BLAST is unable to use the specified file, the statement will set @STATUS to an error code.

Once capture has been enabled, the program must execute one of the following statements before capture begins: TERMINAL, TTRAP, TUPLOAD, or WAIT (with CARRIER or IDLE option). To close the file and save any data that has been captured, use TCAPTURE OFF. The following example shows how a file can be displayed and captured from a remote computer running UNIX:

```
connect
tsend "cat payroll.dat", LF
tcapture ON "payroll.cap" # turn capture on
wait 5 idle # wait for data to stop
tcapture OFF # end capture, close file
```

Chapter 14

Connecting and Disconnecting

Introduction

Connecting and disconnecting are crucial operations. Normally, BLAST initializes the modem and dials a remote system under the control of a specialized modem script called `MODEMS.SCR`. Logging into a remote system, such as a VAX or a UNIX-based computer, is likewise handled by a special script called `SYSTEMS.SCR`. These scripts are called by BLAST when the Connect command is issued from a menu or the `CONNECT` statement is executed in a script. Similarly, disconnecting is managed by `MODEMS.SCR` and `SYSTEMS.SCR`. It is important to understand the structure and operation of these two scripts and how you can modify the scripts.

BLASTscript Libraries

BLAST comes with two script “libraries,” `MODEMS.SCR` and `SYSTEMS.SCR`, that provide the information BLAST needs to con-

trol your modem and to log onto remote computers. These libraries are collections of scripts combined into large files and indexed for rapid access. BLAST automatically chooses the proper scripts from these libraries based on the values of the System Type and Modem Type setup fields. If you should choose to modify either MODEMS.SCR or SYSTEMS.SCR, *be sure to make a backup copy of the file first under another name. As with any other script file, MODEMS.SCR and SYSTEMS.SCR should always be saved as text-only or ASCII files. Do not save them as word-processor files.*

These script libraries are activated through menu commands or script commands, as follows:

Connect – Uses commands in MODEMS.SCR and SYSTEMS.SCR to dial out and log onto the remote system.

Upload – Uses commands in SYSTEMS.SCR to prepare the remote computer for the text upload.

Filetransfer – Uses commands in SYSTEMS.SCR to start BLAST on the remote computer (if the remote computer is a multi-user computer).

Disconnect – Uses commands in MODEMS.SCR and SYSTEMS.SCR to log off the remote system and hang up the modem.

By automating these processes, BLAST allows you to exchange information between many different computer types without requiring technical proficiency in each system.

Modem Control

The MODEMS.SCR library handles a wide range of different modems, some of which may use proprietary commands to perform functions under computer control. BLAST uses the Modem Type setup field or the @MODEM reserved variable to select the proper script from this library and the Originate/Answer setup field or the @ORGANS reserved variable to tell the modem either to originate or to wait for calls.

Remote System Control

The SYSTEMS.SCR library controls the commands sent to the remote computer. By using this library, your PC can start BLAST in host mode on the remote computer. BLAST also uses this library to control text uploading and downloading. BLAST uses the System

Type setup field or the @SYSTYPE reserved variable to select the proper script from this library.

Creating New Libraries

You can create alternate system and modem control files that contain only the necessary commands for your particular hardware—this is more efficient than the standard libraries that include many modems and systems that you are not likely to need. BLAST will always look for individual files in the same directory specified by the BLASTDIR environment variable as MODEMS.SCR and SYSTEMS.SCR before using the standard libraries. For example, if you specify TBLAZER in the Modem Type setup field or set @MODEM to TBLAZER, CONNECT will use a stand alone script named TBLAZER.SCR if it exists to control modem handling instead of the TBLAZER entry in MODEMS.SCR.

The Connection Process in Detail

The MODEMS.SCR library can be used to automate the connection process. If the Modem Type setup field is empty or set to hardware, BLAST assumes that your PC is hardwired to the remote computer and does not open MODEMS.SCR.

When a Modem Type has been selected and the Originate/Answer setup field is set to ANSWER, control is passed to the .ANSWER section in MODEMS.SCR, which initializes the modem and waits for the call.

When the Originate/Answer field is set to ORIGINATE and the Connect command or CONNECT statement is used, control is passed to the .DIAL section. If a phone number is specified in the Phone Number field, .DIAL sends the phone number to the modem as a dial command. If the Phone Number field is empty, .DIAL prompts the user to enter a number. After dialing, it waits for a message from the modem indicating a successful connection has been made.

If a System Type is specified, the corresponding .LOGON section in SYSTEMS.SCR is called for logging onto the remote system. If System Type is empty, BLAST assumes that you do not want system handling and the Connect process ends, returning you to the Online menu or the calling script with @STATUS set to 0.

If an error is detected by MODEMS.SCR or SYSTEMS.SCR, the scripts return to BLAST with @STATUS set to reflect one of the errors listed below:

- 0 No error
- 1 Unable to initialize the modem (MODEMS.SCR)
- 2 No answer (MODEMS.SCR)
- 3 Can't log in: wrong userid, password (SYSTEMS.SCR)
- 4 No Carrier (MODEMS.SCR and SYSTEMS.SCR)
- 5 Busy (MODEMS.SCR)
- 6 No Dialtone (MODEMS.SCR)
- 7 Error (MODEMS.SCR)
- 8 OK unexpected (MODEMS.SCR)

Your script can check @STATUS to determine whether a connection is successful.

The Disconnection Process in Detail

There are four ways to disconnect from another system:

- ◇ You can select Terminal from the Online menu and manually type the appropriate commands to the modem and the remote computer.
- ◇ You can select Disconnect from the Online menu and allow BLAST to automate the process through the SYSTEMS.SCR and MODEMS.SCR libraries.
- ◇ You can write a BLAST script that uses the DISCONNECT statement, which operates similarly to the Disconnect command.
- ◇ You can physically hang up the modem by powering off. This is, of course, not recommended.

The Disconnect process attempts to log off the remote computer using the .LOGOFF section in SYSTEMS.SCR. Control is then transferred to the .HANGUP section in MODEMS.SCR to hang up the modem.

If an error is detected by MODEMS.SCR or SYSTEMS.SCR, the scripts return to BLAST with @STATUS set to reflect one of the errors listed below:

- 0 No error
- 1 Unable to initialize the modem (MODEMS.SCR)
- 3 Can't log out correctly (SYSTEMS.SCR)

Sample Modem Script

The following script illustrates the parts of a modem script. You can incorporate this script into MODEMS.SCR or keep it as a separate file, QUICK.SCR. If you incorporate the script into MODEMS.SCR, you must index the script (see “The Index Utility” on page 199). If you incorporate and index the script, QUICK will appear automatically as a new modem type in the Modem Type setup field. Otherwise, you must enter it manually into the Modem Type setup field.

```
:QUICK
# A sample modem control script illustrating the required
# sections .DIAL, .ANSWER, .HANGUP, and .END.
#
# Dial
.DIAL
  if NULL @PHONENO
    ask "enter phone number", @PHONENO
    if NULL @PHONENO or @STATUS = "-1" return 1
  end
  tsend "ATDT", @PHONENO, CR
  ttrap 45, "CONNECT", "NO CARRIER", "BUSY", "NO DIAL"
  if @STATUS = "1"
    ttrap 2, "\015"
    return 0
  end
  let @STATUS = @STATUS + 2" # set up return code
  return @STATUS
#
# Answer
#
.ANSWER
  tsend "ATS0=1", CR
  ttrap "CONNECT"
  return 0
#
# Hangup
#
.HANGUP
  wait 2
  tsend "+++"
  ttrap 3, "OK", "NO CARRIER"
  if @STATUS = "1"
    wait 1
    tsend "ATH", CR
    ttrap 10, "OK", "NO CARRIER"
    if @STATUS = "0" return 1
```

```

end
wait 1
tsend "ATS0=0", CR
return 0
.END
:
#
# End of QUICK.SCR

```

The required sections for a modem script are `.DIAL`, `.ANSWER`, `.HANGUP`, and `.END`. The appropriate section is activated when the Connect or Disconnect commands are given. The `.END` section terminates the script (or separates the script from the next one in `MODEMS.SCR`) and requires a final colon (:). With this sample, you should be able to write your own modem scripts or modify the scripts in `MODEMS.SCR`. Likewise, you can modify or enhance the system scripts in `SYSTEMS.SCR`.

Slave Script

Dialing into a Computer That is Running `SLAVE.SCR`

Included with the BLAST media is a “slave” script called `SLAVE.SCR`, which keeps BLAST continually ready to accept any incoming calls from a computer running BLAST protocol. When you dial into a computer running `SLAVE.SCR` and then enter Filetransfer, the remote computer becomes a “slave” to your computer. While the remote computer is in slave mode, you have access to all Filetransfer menu options.

NOTE: After you have connected to the computer running `SLAVE.SCR`, you have 25 seconds to select Filetransfer from the Online menu. If Filetransfer is not selected within this time, the slave assumes that the call is not for BLAST, hangs up the modem, and resets for the next call. Also note, if you exceed the Inactivity Timeout while performing local commands via the Filetransfer Local menu option, you will be disconnected.

To end a “slave” session, simply exit Filetransfer mode.

Setting Up BLAST to Run in Slave Mode

If you choose to setup your computer to accept incoming calls and run in slave mode, you must follow these steps:

- ◇ Set the Connection setup field to the modem port.
- ◇ Set the Protocol setup field to BLAST.

- ◇ From the Offline menu, choose Online (do *not* choose Connect).
- ◇ From the Online menu, choose Script.
- ◇ At the prompt for a script filename, type “slave” (the “.scr” extension is not needed).
- ◇ Press ENTER.

BLAST will then be set to receive incoming calls and to begin slave mode when the remote user enters Filetransfer. When the remote user disconnects, SLAVE.SCR resets BLAST to accept the next incoming call.

The Index Utility

Three files used by BLAST contain an index at the beginning of the file: BLAST.HLP, MODEMS.SCR, and SYSTEMS.SCR. Each index contains references to specific sections in the file. For instance, MODEMS.SCR contains a BLASTscript section to control the US Robotics Courier modem. The index at the beginning of MODEMS.SCR contains a reference to this section.

Indexing a file allows BLAST to jump to a particular section of a file quickly. Each section of the file should begin with a label in the form:

:LABEL

The index itself is in the form of lines of text, each beginning with the greater-than sign (>). The Index utility adds the numeric references that send control to the referenced section of the file.

If you modify any of these files, the index must be recalculated so that BLAST can read the file properly. For example, if you add a new system type to SYSTEMS.SCR or add your own Online Help text to BLAST.HLP, you must run the Index utility copied to your directory during installation to re-index the file. Indexing should only be performed on these three files. Before modifying or re-indexing any of these files, however, *be sure to make a backup copy of the file under another name and save the file you are modifying as text-only or ASCII.*

If you create a separate modem script, such as MYMODEM.SCR, and enter MYMODEM as the Modem Type in a setup, indexing is not

required. If you modify any of the three standard files, however, you must re-index them. Follow this procedure to index a file:

1. Make a backup copy of the original file under another name.
2. Make the required changes to the original file.
3. Save the file as text-only.
4. Rename the modified file.
5. Type the following command:

```
index oldfile newfile
```

where *oldfile* is the modified file and *newfile* is the name of the new indexed file. For example, if you modified SYSTEMS.SCR and saved it under the name SYS.SCR, you would type the following:

```
index sys.scr systems.scr
```

Remember also that BLAST will not operate properly if the final name of the file is not exactly as described above, that is, either SYSTEMS.SCR, MODEMS.SCR, or BLAST.HLP.

Chapter 15

BLASTscript Command Reference

Introduction

As you learned in Chapter 12, BLAST's script commands are English-like statements that automate communications functions. This chapter defines and illustrates the use of BLAST's script commands.

To use the script commands correctly, you must understand the data types supported by BLASTscript and the syntax rules defining a legal script statement.

Data Types

All data is stored as strings that may be up to 139 characters in length.

Variables

Variables start with “@”, followed by up to eight unique characters and any additional characters. For example:

```
@X
@Fred
@123
@logdateformat
```

Names are not case-sensitive. Thus @Fred, @fred, and @FRED all refer to the same variable.

Numeric Constants

Numeric constants are sequences of digits enclosed in double quotation marks. They may not be preceded by a minus sign. For example:

```
" 4 "
" 4789 "
" 56 "
```

Numeric Strings

Numeric strings are sequences of digits enclosed in double quotation marks. Numeric strings may be preceded by a minus sign. For example:

```
" -4 "
" 4789 "
" -56 "
```

Numeric Values

Numeric values may be variables, numeric constants, or numeric strings as defined above.

String Constants

String constants are alpha-numeric sequences enclosed in double quotation marks. For example:

```
"THIS IS A STRING CONSTANT"
"12345"
"123ABC"
```

String constants may contain special control characters:

<code>\r</code>	carriage return
<code>\l</code>	linefeed
<code>\f</code>	formfeed
<code>\b</code>	backspace
<code>\t</code>	tab
<code>\\</code>	backslash character
<code>\xxx</code>	where xxx is the three-digit octal value of the character except for the octal value of null (<code>\000</code>), which is not permitted because null characters are treated as end-of-string characters. When encountered, nulls stop string processing.

Specifically, keep the backslash character in mind in writing scripts. If you quote a pathname, you will need to use double backslashes, as in the following example:

```
set @mydir = "\\DOS\\cih"
filetransfer
send
cih
@mydir

esc
```

If you want to include quotation marks in a `DISPLAY` or `WRITE` statement, a backslash must precede the quotation marks; otherwise, `BLAST` interprets the second quotation mark as the end of the string. For example, to display the following

```
Processing "Weekly Reports" -- please wait.
```

your script statement would be:

```
display "Processing \"Weekly Reports\" -- please wait."
```

Control characters may be coded in a string by preceding the character with “`^`”. For example, `^M` is equivalent to `\r` and `\015`:

```
set @msg = "3 carriage returns: ^M, \r, \015"
```

To code a single `^` in a string, two `^` characters are coded together.

String Values

String values may be string constants or variables as defined above.

Reserved Variables

Reserved variable values correspond to setup fields and physical or logical program conditions. See Chapter 16 for more information.

Syntax Rules

The number of characters in a script statement is limited to 131 characters. Indentation makes code easier to read and has no effect on operation. Commands and variable names are not case-sensitive. Thus,

```
SET @FILENAME = "C:\\BLAST\\default.su"
```

is equivalent to

```
set @filename = "C:\\BLAST\\default.su"
```

When strings are numeric values, mathematical operations (+, -, *, /) can be performed in a LET statement. Parentheses are *not* allowed, however, and expressions are evaluated left to right without precedence.

Comment lines begin with “#”. Comments may also be placed on the same line as a BLASTscript statement by putting a # in the line; all characters from the # to the end of the line are treated as a comment.

Every line in a script must be executable or contain a comment. As a consequence, blank lines, which are rarely executable, cannot be used to separate script code visually.

BLASTscript is highly space-sensitive. When in doubt, separate all elements of a statement with spaces and enclose all constants, strings, or numerals in quotation marks. For example:

```
set @variable = "hello, world"
```

Commands That Set @STATUS

A number of script commands set the value of @STATUS, indicating whether the command was executed successfully. In general, @STATUS is set to 0 to indicate success. Some commands that re-

turn numeric results (e.g., STRINX, TTRAP) set @STATUS to 0 to indicate a null condition. The following commands set @STATUS:

ASCII	FOPENR	LPRINT	STRLEN
ASK	FOPENW	LRENAME	TCAPTURE
CALL	FREAD	LTYPE	TSEND
CONNECT	FREWIND	NEW	TTRAP
DISCONNECT	FWRITE	RAISE	TUPLOAD
DROP	LCHDIR	REMOVE	WAIT CARRIER
FCLOSE	LDELETE	RETURN	WAIT IDLE
FILETRANSFER	LLIST	SELECT	
FOPENA	LOAD	STRINX	

Setup Reserved Variables and @STATUS

@STATUS is also set by the following commands when the commands assign a value to a reserved variable associated with a setup field: ASK, FREAD, LET, LOWER, SET, STRTRIM, and UPPER.

BLASTscript Statements

This section is organized alphabetically by command. The following conventions are used throughout:

- [] Indicates that enclosed phrases or characters are optional.
- ... Indicates that the preceding statement or line may be repeated.
- { xx | yy } Indicates that either the xx or yy phrase is required. Choose only one.

ASCII

get ASCII value of a character

FORMAT: *ASCII_string_value, numeric_value*

ASCII sets @STATUS to the ASCII value of the character at position *numeric_value* within *string_value*. The first position is 1. The ASCII value is the decimal value given to the ASCII character. For these values, see Appendix D.

EXAMPLE:

```
set @filename = "\\path\\filename"
ASCII @filename, 1          # get ASCII code for the first
                           # character in @filename
if @STATUS = "92"         # ASCII 92 is a backslash (\)
    display @filename, "is a full pathname"
end
```

ASK

prompt for a string from the user

FORMAT: **ASK** [NOECHO] *string_value*, *variable*

ASK prompts the user with the *string_value* displayed at the top left of the screen. The input from the user will be placed in *variable*. Because of display limitations, the combined length of *string_value* and *variable* should not exceed 80 characters. The NOECHO option causes BLAST to suppress user input. Use the NOECHO option when entering a password or other sensitive data.

If the input ends with ENTER, @STATUS will be set to 0. If ESC is pressed instead of ENTER, @STATUS will be set to a nonzero value *unless* the variable in the ASK statement is a reserved variable; in this case, the behavior of ASK is undefined. For this reason, we strongly recommend that you *not* use reserved variables in an ASK statement.

EXAMPLE:

```
ASK "what month", @month
ASK NOECHO "Password?", @secret     # no display
```

CALL

call another script

FORMAT: **CALL** *string_value*

CALL loads and executes another BLAST script, after which the called script returns to the calling script. *String_value* contains the filename of the called program. If the script is successfully called, @STATUS is set to 0; if the called script does not exist, @STATUS is set to 1.

On return from the called script, @STATUS is set to the value of the exit code in the called program's RETURN statement or to 0 if no exit code value is given. Since all values are global, any values set

in the calling script will be retained in the called script and vice versa. CALL searches for the script name in the following order:

1. Files without “.SCR” extension in current working directory.
2. Files with “.SCR” extension in current working directory.
3. Files without “.SCR” extension in BLASTDIR directory.
4. Files with “.SCR” extension in BLASTDIR directory.

IMPORTANT: When writing a script that will be CALLED, you may *not* want to assign a RETURN value of 1 if the calling script checks @STATUS to determine the success of CALL. Since @STATUS is set to 1 when a CALLED script cannot be found, assigning a value of 1 in the RETURN statement would invalidate the check of CALL.

EXAMPLE:

```
CALL "backup.scr"  
if @STATUS = "0" display "Backup Successful"
```

CLEAR

clear the scrolling region

FORMAT: *CLEAR*

CLEAR clears the scrolling region of the screen. This command affects only script operations, not terminal emulation.

EXAMPLE:

```
CLEAR
```

CLEOL

clear to the end of the line

FORMAT: *CLEOL*

CLEOL clears from the current cursor position to the end of the current line in the scrolling region. This command affects only script operations, not terminal emulation.

EXAMPLE:

```
CLEOL
```

CONNECT

connect to a remote

FORMAT: *CONNECT*

CONNECT directs BLAST to execute routines in the MODEMS.SCR and SYSTEMS.SCR libraries to dial the modem and log on if the Modem and System Type setup fields are specified. If CONNECT is successful, @STATUS is set to 0. For more information about the operation of the CONNECT command, see Chapter 14.

EXAMPLE:

```
CONNECT
if @STATUS = "0" display "OK"
```

CURSOR

position the cursor within the scrolling region

FORMAT: *CURSOR numeric_value1, numeric_value2*

CURSOR positions the cursor to a given row (*numeric_value1*) and column (*numeric_value2*) in the 20 x 80 scrolling region. The row ranges from 0 to 19, column from 0 to 79. If @USERIF is set to 0 or OFF, the full 24 x 80 screen will be addressed.

Use PUT statements following cursor position to write on the screen.

EXAMPLE:

```
CURSOR 4, 10                            # move to row 4, column 10
put "1. Get sales figures"
CURSOR 6, 10
put "2. Send pricing"
ask "enter option (1 or 2)", @opt
```

DISCONNECT

disconnect from a remote

FORMAT: *DISCONNECT*

DISCONNECT directs BLAST to execute routines in SYSTEMS.SCR and MODEMS.SCR to log off and hang up the modem if the System and Modem Type setup fields are specified. If DISCONNECT is successful, @STATUS is set to 0. See Chapter 14 for a full discussion.

EXAMPLE:

```
DISCONNECT
if @STATUS = "0" display "OK"
```

DISPLAY

display strings to display region

FORMAT: *DISPLAY string_value, ...*

DISPLAY displays messages in the scrolling region of the screen. If a log file has been specified, these messages will also be sent to the log file.

EXAMPLE:

```
DISPLAY "Dialing...", @PHONENO
```

DROP

drop DTR / RTS

FORMAT: *DROP {DTR | RTS }*

DROP terminates signals on the RS-232 interface. If the value is DTR, the Data-Terminal-Ready signal drops, hanging up most modems (cable and modem configuration permitting). If the value is RTS, the Request-to-Send signal drops, causing some devices to stop transmitting. DROP DTR and DROP RTS commands are supported for standard internal (non-multiplexor) ports only. If DROP DTR or DROP RTS is successful, @STATUS is set to 0.

EXAMPLE:

```
DROP DTR        # drop DTR signal
DROP RTS        # drop RTS signal
```

ECHO

enable/disable script display

FORMAT: *ECHO {ON | OFF }*

ECHO ON traces BLASTscript statements, displays them on the screen as they are executed, and writes them to a log file if one is specified. When executing CONNECT and DISCONNECT statements, the statements in the MODEMS.SCR and SYSTEMS.SCR libraries are also echoed. If you do not wish to see all these statements, turn ECHO ON only as needed.

Because the statements displayed by ECHO are interspersed with the standard interactive dialog, ECHO is particularly useful in understanding what activity is triggered by what response within a BLAST script.

EXAMPLE:

```
ECHO ON      # set echo on
ECHO OFF     # set echo off
```

ERRSTR

store script error text

FORMAT: *ERRSTR numeric_value, string_variable*

ERRSTR puts the English language error message corresponding to *numeric_value* in *string_variable*. This statement is commonly used in association with the reserved variable @SCRIPTERR, which contains the number of the last BLASTscript error encountered.

For a list of error messages, see Appendix A. Note that not all error messages listed are possible errors in all versions of BLAST; some are operating system specific.

EXAMPLE:

```
fopenr 1, "nonexist.fil"
if @STATUS not = "0"
  ERRSTR @SCRIPTERR, @MESSAGE
  display "ERROR #", @SCRIPTERR, "-", @MESSAGE
end
```

FCLOSE

close an open file

FORMAT: *FCLOSE_numeric_constant*

FCLOSE closes an open file. *Numeric constant* is a number, called a handle, that other file statements use to refer to the file. The file handle can range from 1 to the number of file handles available through the operating system. If FCLOSE is successful, @STATUS is set to 0.

EXAMPLE:

```
fopenr 1, "input.fil"           # open file 1 for reading
FCLOSE 1                        # close file 1
```

FILETRANSFER FILE

perform commands from a BLAST TCF

FORMAT: *FILETRANSFER*
 FILE
 filename
 ESC

In BLAST protocol, this multi-line statement performs commands read from a transfer command file (TCF). *Filename* is the transfer command file, which may be specified with a string variable. See “Transfer Command File” on page 117 for a complete description of the transfer command file format.

EXAMPLE:

```
connect
FILETRANSFER
FILE
command.fil
ESC
disconnect
quit
```

FILETRANSFER GET / SEND

get/send file

FORMAT: *FILETRANSFER* *FILETRANSFER*
 GET *SEND*
 {protocol-dependent string(s) ...} *{protocol-dependent string(s) ...}*
 ESC *ESC*

These statements transfer files to and from the remote computer. The exact syntax is protocol-dependent. For a full description of the syntax of the individual protocols, see “File Transfers with BLAST Session Protocol” on page 178 and the sections on scripting file transfers for the other supported protocols in Chapter 13.

EXAMPLE:

```
set @protocol = "BLAST"
set @new = "usr\\blast\\readme"
FILETRANSFER                    # enter Filetransfer mode
GET                                # get a file with BLAST
getme.fil                         # remote filename
@new                               # local filename stored in a variable
to                                 # text conversion and overwrite
SEND                               # send a file with BLAST
```

```

*.DOC          # may be lots of files
%             # resolve multiple names with %

SEND          # send a file with no remote filename
samename.fil  # this will also be the remote name

t            # send as text file
ESC         # end BLAST protocol session

```

FILETRANSFER LOCAL

perform local commands using BLAST protocol

FORMAT:

```

FILETRANSFER
LOCAL
{LIST          | DELETE | RENAME | TYPE | PRINT | CHDIR | SYSTEM}
{SHORT|LONG} filename oldname filename filename pathname command
filename      ESC      newname  ESC   ESC   ESC   ESC
ESC          ESC      ESC      ESC   ESC   ESC   ESC
ESC          ESC      ESC

```

This multi-line statement performs Local menu commands within a FILETRANSFER-ESC block using BLAST protocol. Note that Local menu commands may also be performed with the LLIST, LDELETE, LPRINT, LTYPE, LRENAME, and LCHDIR statements.

LOCAL is followed by one or more commands. Most of the commands are followed by a filename, which may include wildcards or a string variable. Please note that lengthy local functions may force either the remote system or your system to time out, so keep local functions as short as possible or change the Inactivity T/O setup field to allow more time.

LIST – Display your local directory listing. The line after LIST must specify either SHORT or LONG. The second line after LIST can be left blank to display all files or it can be a filename, which may include wildcards (e.g., *.TXT).

DELETE – Delete a file or files on your system. The line following DELETE is the filename, which may include wildcards.

RENAME – Rename a file on your system. The line after RENAME is the old filename; the second line after RENAME is the new filename.

TYPE – Type a file on your system's display. The line following TYPE is the filename.

- PRINT** – Print a file on your system’s default printer. The line following **PRINT** is the filename.
- CHDIR** – Change the working directory of your system. The line following **CHDIR** is the pathname of the new working directory.
- SYSTEM** – Perform a local system command. The line following **SYSTEM** is a system command. If this line is left blank, **BLAST** invokes the operating system interactively. When you are finished with the command interpreter, you must return to **BLAST** by typing `exit` and pressing **ENTER**. When **BLAST** is started with the `/b` switch (or with the `/n` switch if the display has not been re-enabled through a script), you cannot escape to a system prompt (see “Command Line Switches” on page 10).

EXAMPLE:

```
set @protocol = "BLAST"
FILETRANSFER          # start BLAST session protocol
GET
daily.dat
new.dat
to
LOCAL                  # begin LOCAL commands
  PRINT
  new.dat
  RENAME
  new.dat
  old.dat
  ESC                  # end LOCAL commands
SEND
sendme.fil
toyou.fil
t
ESC                    # end BLAST protocol session
```

FILETRANSFER MESSAGE

send messages using **BLAST** Protocol

FORMAT: **FILETRANSFER**
MESSAGE
message
ESC

Using **BLAST** protocol, **MESSAGE** sends a text string that is displayed in the scrolling region of both computers’ displays. The line

after MESSAGE is a message—a line of text up to 67 characters or a variable containing a line of text up to 67 characters.

EXAMPLE:

```
FILETRANSFER          # enter Filetransfer mode
MESSAGE               # send a message
Sending Sales Reports # specify the message
ESC
```

FILETRANSFER REMOTE

perform remote commands

FORMAT for BLAST protocol:

```
FILETRANSFER
REMOTE
{LIST | DELETE | RENAME | TYPE | PRINT | CHDIR | MORE}
{SHORT|LONG} filename oldname filename filename pathname ESC
filename ESC newname ESC ESC ESC ESC ESC
ESC ESC ESC ESC ESC ESC
ESC ESC
```

This multi-line statement performs error-free file management on the remote computer during BLAST session protocol. Multiple commands may follow the REMOTE command, and filenames (valid pathnames for the remote computer) or string variables may follow each command. Some older versions of BLAST do not support REMOTE commands. During a BLAST protocol session, the following commands are available:

LIST – Display the remote directory listing. The line after LIST must specify either SHORT or LONG. The second line after LIST can be left blank to display all files or it can be a filename, which may include wildcards (e.g., *.TXT).

DELETE – Delete a file or files on the remote system. The line following DELETE is the filename, which may include wildcards.

RENAME – Rename a remote file. The line after RENAME is the old filename; the second line after RENAME is the new filename.

TYPE – Type a remote file on your system's display. The line following TYPE is the filename.

PRINT – Print a remote file to the remote system default printer.
The line following PRINT is the filename.

CHDIR – Change the working directory on the remote computer.
The line following CHDIR is the pathname of the new
working directory.

MORE – Continue displaying data from the remote computer after a
page pause.

FORMAT for Kermit server protocol:

FILETRANSFER

REMOTE

{DIRECTORY	ERASE	TYPE	CWD	SPACE	WHO	MESSAGE	HOST	KERMIT	HELP}
pathname	filename	filename	pathname	pathname	user	message	command	message	ESC
password	ESC	ESC	ESC	ESC	ESC	ESC	ESC	ESC	ESC
ESC	ESC	ESC	ESC	ESC	ESC	ESC	ESC	ESC	
ESC									

During a Kermit server protocol session, the available commands depend upon both the version and the configuration of the remote Kermit server. A command may fail if the remote Kermit server does not support the command. You must start Kermit remote server on the remote system before entering Kermit Filetransfer mode.

Kermit remote commands include:

DIRECTORY – Display a directory on the remote server. The line after DIRECTORY is the pathname (with or without wildcards) of the remote directory for which you want a listing; if you leave this line blank, the current working directory listing of the remote server will be displayed. The second line after DIRECTORY is the password that may be required to gain access to the directory listing. If no password is required, leave this line blank.

ERASE – Delete a file on the server. The line following ERASE is the filename (with or without wildcards) of the file to be erased. If you do not specify a full path for the file, the file (if it exists) will be removed from the current working directory of the remote server.

TYPE – Display a remote-server file on your screen. The line following TYPE is the filename of the file to be displayed. Kermit does not support a page pause, so you must use CTRL S to pause and CTRL Q to resume the flow of data.

CWD – Change the server's working directory. The line following CWD is the pathname of the new working directory.

- SPACE** – Display unused drive space of a directory on the remote server. The line following **SPACE** is the pathname (with or without wildcards) of the directory for which unused drive space is to be reported.
- WHO** – Display information on user(s) currently logged onto the server. The line following **WHO** is the user for whom you want information. If you leave this line blank, information on all users logged onto the server will be displayed.
- MESSAGE** – Send a one-line message to be displayed to the remote operator. The line following **MESSAGE** is the one-line message to be displayed to the remote operator.
- HOST** – Send an operating system command to the server. The line following **HOST** is the operating system command to be sent to the remote server. The command is executed immediately.
- KERMIT** – Send a Kermit language command to modify session parameters. The line following **KERMIT** is the message (Kermit language command) to be issued to the Kermit server, for example, **SET FILE TYPE BINARY**.
- HELP** – Display a short list of the available commands on the server.

EXAMPLE:

```

tsend "kermit -x", LF      # start kermit server on remote
FILETRANSFER             # enter Filetransfer mode
get
daily.dat
new.dat
REMOTE                   # start REMOTE commands
  CWD
  /usr/customer
  TYPE
  contactlist.txt
  ESC                     # end REMOTE commands
send
sendme.fil
toyou.fil
ESC                       # end Kermit protocol session

```

FLUSH

clear the input buffer

FORMAT: *FLUSH*

FLUSH clears the communications port input buffer. Only characters received after the FLUSH command has been executed will be available.

EXAMPLE:

```
FLUSH                            # empty buffer
ttrap 10, "@"                   # trap for "@"
```

FOPENA

open a file for appending

FORMAT: *FOPENA numeric_constant, string_value*

FOPENA opens a file for appending. If the file does not exist, it will be created. If it does exist, it will be opened and subsequent writes will append data to the end of the file. *String_value* is the filename of the file to be opened. *Numeric_constant* is a number, called a handle, that other file statements use to refer to the file. The file handle can range from 1 to the number of file handles available through the operating system. If FOPENA is successful, @STATUS is set to 0.

EXAMPLE:

```
FOPENA 1, "script.log"            # open file 1 for appending
fwrite 1, "got this far"         # adds string to the file
fclose 1                          # close file 1
```

FOPENR

open a file for reading

FORMAT: *FOPENR numeric_constant, string_value*

FOPENR opens a file for reading. The file must already exist. *String_value* is the filename of the file to be opened. *Numeric_constant* is a number, called a handle, that other file statements use to refer to the file. The file handle can range from 1 to the number of file handles available through the operating system. If FOPENR is successful, @STATUS is set to 0.

EXAMPLE:

```
FOPENR 1, "command.fil"      # open file 1 for reading
fread 1, @input              # read the first line
fclose 1                     # close file 1
```

FOPENW

open a file for writing

FORMAT: *FOPENW numeric_constant, string_value*

FOPENW opens a file for writing. If the file does not exist, it will be created. If it does exist, all data in the file will be overwritten. *String_value* is the filename of the file to be opened, and *numeric_constant* is a number, called a handle, that other file statements use to refer to the file. The file handle can range from 1 to the number of file handles available through the operating system. If FOPENW is successful, @STATUS is set to 0.

EXAMPLE:

```
FOPENW 1, "cscript.log"      # open file 1 for writing
fwrite 1, "got this far"     # write string to file 1
fclose 1                     # close file 1
```

FREAD

read a line from a file

FORMAT: *FREAD numeric_constant, variable*

After an FOPENR command, FREAD reads a line of text into *variable*. *Numeric_constant* is the file handle assigned the file in the FOPENR statement. If FREAD is successful, @STATUS is set to 0. A nonzero value indicates an error reading the file or end of file. If the variable in an FREAD statement is a reserved variable, the behavior of FREAD is undefined. For this reason, we strongly recommend that you *not* use reserved variables in an FREAD statement.

EXAMPLE:

```
fopenr 1, "command.fil"      # open file 1 for reading
FREAD 1, @input              # read line into @input
if @STATUS not = "0"
  display "End of file reached"
end
fclose 1                     # close file
```

FREE

release a variable from memory

FORMAT: *FREE* user-defined variable

FREE releases memory allocated to the specified user-defined variable. To recover all memory, you must *FREE* variables in the reverse order in which they were defined.

EXAMPLE:

```
FREE @input
```

FREWIND

rewind a file

FORMAT: *FREWIND* numeric_constant

FREWIND “rewinds” a file by resetting the file pointer to the beginning of the file. *Numeric_constant* is the file handle assigned the file in an *FOPENR*, *FOPENW*, or *FOPENA* statement. If *FREWIND* is successful, *@STATUS* is set to 0.

EXAMPLE:

```
fopenr 1, "commands.fil"        # open file 1 for reading
fread 1, @input                # read first line of file 1
FREWIND 1                      # rewind file 1
fread 1, @also                # read first line again
fclose 1                        # close file 1
```

FWRITE

write a line to a file

FORMAT: *FWRITE* numeric_constant, string_value,...

After an *FOPENW* command, *FWRITE* writes out a series of one or more strings to a file as a single line of text. *Numeric_constant* is the file handle assigned the file in an *FOPENW* or *FOPENA* statement. If *FWRITE* is successful, *@STATUS* is set to 0.

EXAMPLE:

```
fopenw 1, "output.fil"
FWRITE 1, "the userid is: ", @USERID
fclose 1
```

GOTO

branch to another point in program

FORMAT: *GOTO .LABEL*

GOTO branches unconditionally to another location in the program. GOTO will abort the program if *.LABEL* cannot be found. The label is not case-sensitive and consists of eight characters or less, not counting the initial period.

EXAMPLE:

```
.PWD
ask "enter the secret word", @pword
if @pword = "rosebud" GOTO .CONT
werror "invalid name"
GOTO .PWD
.CONT
display "Good morning, Mr. Phelps"
```

IF

perform single action if condition is true

FORMAT: *IF condition [{and / or}...] statement*

IF performs *statement* when *condition* is true. Evaluation is from left to right. Parentheses and arithmetic functions are not permitted in the condition. The syntax of *condition* can be one of two forms. The first form is valid for string values only:

string_value1 [NOT][>|>=|<|<=|=] *string_value2*

The condition is true when *string_value1* is:

- > greater than
- >= greater than or equal to
- < less than
- <= less than or equal to
- = equal to

string_value2.

The comparison is based on the ASCII values and the length of the strings. If the strings are not equal, the comparison is performed on the first different character in the strings.

The second form of the conditional clause is valid for numeric values only:

```
numeric_value1 [NOT ][[GT|GE|LT|LE|EQ] numeric_value2
```

The condition is true when *numeric_value1* is:

GT greater than
GE greater than or equal to
LT less than
LE less than or equal to
EQ equal to

numeric_value2.

Some special qualifiers provide an implied *condition*:

[NOT]NULL *string_value*
True [False] when *string_value* is of zero length.

[NOT]*numeric_constant*
True [False] when *numeric_constant* equals @STATUS.

[NOT]REPS
True [False] when the REPS counter is not zero (see page 170 for more information on using REPS and loops).

[NOT]EXIST *string_value*
True [False] when a file named the value of *string_value* exists.

[NOT]OK
True [False] when @STATUS = "0".

EXAMPLE:

```
IF EXIST "file.one" LDELETE "file.one"  
IF NOT NULL @VAR Display "@VAR is not empty"  
IF @USERID = "FRED" GOTO .SENDFILES
```

The following three statements are all equivalent:

```
IF OK GOTO .RUN  
IF @STATUS = "0" GOTO .RUN  
IF 0 GOTO .RUN
```

IF – ELSE

perform action for true or false conditions

FORMAT: *IF condition [{and / or}...] statement*
 ELSE statement

IF-ELSE performs *statement* based upon *condition*. When the condition is true, the statement following the condition executes. When condition is false, the statement after ELSE executes. Statement must be on the same line as condition.

EXAMPLE:

```
connect
IF @STATUS = "0" write "Logged on successfully."
ELSE write "Logon failed!"
```

IF – END

perform multiple actions if condition is true

FORMAT: *IF condition [{and / or} condition...]*
 statement
 END

This multi-line clause performs several *statements* based upon *condition*. When the *condition* is true, subsequent *statements* up to the END are executed.

EXAMPLE:

```
IF @USERID NOT = "Annie"
    display "You can't run this script!"
    return 2
END
```

IF – END / ELSE – END

perform several actions for true or false conditions

FORMAT: *IF condition [{and / or} condition...]*
 statement
 END
 ELSE
 statement
 END

This multi-line clause performs several *statements* based upon *condition*. When the *condition* is true, the *statements* up to the first

END are executed. When the *condition* is false, the *statements* following ELSE and up to the END are executed.

When execution speed is important, use this statement instead of GOTO. Also, programs using this programming structure are generally easier to understand and maintain than programs using GOTO.

EXAMPLE:

```
ask "Ok to Log on?", @answer
IF @answer = "YES"
    display "Now Logging on"
    tsend @USERID, CR
END
ELSE
    display "Will not attempt to Log on"
    tsend "BYE", CR
END
```

LCHDIR

change working directory

FORMAT: *LCHDIR string_value*

LCHDIR changes the current working directory on the local computer to the directory specified in the *string_value*. Note that you must use double backslashes inside of quotation marks. If LCHDIR is successful, @STATUS is set to 0.

EXAMPLE:

```
LCHDIR "\\work"                    # change directory to \work
if @STATUS = "0"                   # if the return status is 0
    display "CHDIR ok"             # success!
end
```

LDELETE

delete a file on the local system

FORMAT: *LDELETE string_value*

LDELETE deletes from the local computer the file specified in *string_value*. If LDELETE is successful, @STATUS is set to 0.

EXAMPLE:

```
LDELETE "sales.jun"
if @STATUS = "0" display "sales.jun deleted"
```

LET

perform simple arithmetic

FORMAT: *LET variable = numeric value [{+ | - | * | /} numeric value]..*

LET does simple integer arithmetic. The expression is evaluated from left to right, with no grouping or precedence. The result is placed in variable. The maximum and minimum integer values are 32,767 and negative 32,768, respectively. If the variable in a LET statement is a reserved variable, the behavior of LET is undefined. For this reason, we strongly recommend that you *not* use reserved variables in an LET statement.

EXAMPLE:

```
display "Polling statistics:"
LET @total = @numbad + @numgood
display "Total sites polled: ", @total
LET @next = @next + "1"
display "Next site is site number: ", @next
```

LLIST

display a listing of files on the system

FORMAT: *LLIST [LONG] string_value*

LLIST displays a directory listing on the local computer as specified by *string_value*. Wildcards may be used. If no path is given, items from the local current directory are listed. If LONG is specified, the listing will give all accompanying data rather than just the filenames and directory names. If the LLIST is successful, @STATUS is set to 0.

EXAMPLE:

```
LLIST LONG "*.BAT"                    # long listing of file info
if @STATUS = "0" display "list OK"
```

LOAD

load a system setup

FORMAT: *LOAD string_value*

LOAD loads a setup from the directory specified by the SETUPDIR environment variable. *String_value* is the name of the setup. The setup name should not include the .SU extension. This statement operates like the Offline menu Select command and the SELECT

statement. If the setup has been successfully loaded, @STATUS is set to 0.

EXAMPLE:

```
LOAD "Blaster"  
if @STATUS = "0"  
    display "Setup Blaster is the current setup"  
end  
else  
    display "can't load the setup Blaster"  
end
```

LOCAL SYSTEM

perform operating system command

FORMAT: *LOCAL*
 SYSTEM
 string_value
 ESC

This multi-line statement performs local operating system commands. The line following SYSTEM is the system command. If this line is left blank, BLAST invokes the operating system interactively. When you are finished with the command interpreter, you must return to BLAST by typing *exit*. Because of memory constraints, we do not recommend escaping to a system prompt and executing another program.

EXAMPLE:

```
set @syscmd = "dir /w > catalog.txt"  
LOCAL  
SYSTEM  
COPY A:FRED B:WILMA  
SYSTEM  
@syscmd  
ESC
```

LOWER

convert variable to lowercase

FORMAT: *LOWER variable*

LOWER changes all uppercase characters in a variable to lowercase. If the variable in a LOWER statement is a reserved variable, the behavior of LOWER is undefined. For this reason, we strongly recommend that you *not* use reserved variables in a LOWER statement.

EXAMPLE:

```
ask "Enter your name:", @name  
LOWER @name
```

LPRINT

print a file on the local printer

FORMAT: *LPRINT string_value*

LPRINT prints the file specified by *string_value* to the local printer. If LPRINT is executed, @STATUS is set to 0. For information on how to specify the printer used, see Online Help for the Local menu Print command.

EXAMPLE:

```
LPRINT "salesdata"  
if @STATUS = "0" display "print worked ok"
```

LRENAME

rename a file on the local system

FORMAT: *LRENAME string_value1, string_value2*

LRENAME renames the local file specified in *string_value1* to the name specified in *string_value2* on the local computer. If LRENAME is successful, @STATUS is set to 0.

EXAMPLE:

```
LRENAME "f1.dat", "f2.dat"  
if @STATUS = "0" display "Rename worked"
```

LTYPE

type a file on the local screen

FORMAT: *LTYPE string_value*

LTYPE types the local file specified in *string_value* on the screen. If the LTYPE is successful, @STATUS is set to 0.

EXAMPLE:

```
LTYPE "salesdata"                                    # display salesdata  
if @STATUS = "0" display "LTYPE worked"
```

MENU

enable/disable menu display during script execution

FORMAT: *MENU {ON | OFF}*

MENU ON leaves the menu displayed for debugging purposes while a BLAST script is executing. Normally, menu display is suppressed during script execution.

EXAMPLE:

```
MENU ON        # set the menu display on
```

NEW

create a new BLAST setup

FORMAT: *NEW string_value*

NEW creates a new setup in the directory specified by the SETUPDIR environment variable (see page 9), based on the current values in memory. *String_value* is the name of the setup. The setup name should not include the .SU extension.

The NEW statement operates like the Offline menu New command. If you specify a setup name that already exists, NEW will load that setup instead of creating a new one. If the setup has been successfully created, @STATUS is set to 0.

NOTE: Before creating a new setup, the user should first check to see if a setup with the same name already exists. The user can do this with an IF EXIST statement as shown in the example below:

EXAMPLE:

```
if exist "CI.SU"
  display "Setup with that name already exists."
end
else
  NEW "CI"                                # create setup named ci.su
  if ok display "New setup created."
  else display "Couldn't create new setup."
end
```

PUT

output strings to the scrolling region

FORMAT: *PUT string_value,...*

PUT outputs one or more strings to the scrolling region. There is no implicit carriage return or new line after the output. This command is usually used in conjunction with the CURSOR statement.

EXAMPLE:

```
cursor 9, 30                            # put cursor in row 9,col 30
PUT "The winner is ", @win            # display string at
                                      # cursor position
```

QUIT

quit BLAST and return to system with exit code

FORMAT: *QUIT numeric_constant*

QUIT aborts BLAST and returns to the operating system. *Numeric_constant* is an exit code that can be tested by the operating system.

EXAMPLE:

```
QUIT 123                                # exit to operating system, exit status 123
```

RAISE

raise DTR/RTS

FORMAT: *RAISE {DTR | RTS}*

RAISE raises the Data-Terminal-Ready signal (DTR) or the Request-to-Send signal (RTS) on the RS-232 interface. These signals are normally used with modems. Some systems have DTR and RTS tied together so that raising either one affects both signals. RAISE DTR and RAISE RTS commands are supported for standard internal (non-multiplexor) ports only. If RAISE DTR or RAISE RTS is successful, @STATUS is set to 0.

EXAMPLE:

```
RAISE DTR                              # raise the DTR signal
RAISE RTS                              # raise the RTS signal
```


REMOVE

remove a system setup

FORMAT: *REMOVE string_value*

REMOVE deletes a setup from the directory specified by the SETUPDIR environment variable. *String_value* is the name of the setup. The setup name should not include the .SU extension. If the setup has been successfully removed, @STATUS is set to 0.

EXAMPLE:

```
REMOVE "blaster"                                # delete blaster.su
if @STATUS = "0" display "Setup Blaster has been removed."
```

REPS

set repetition counter

FORMAT: *REPS numeric_value*

REPS creates loops in BLAST scripts. When REPS is used in an IF statement, it keeps track of the number of repetitions performed. The REPS numeric value is decremented and then tested for a value of zero. If *numeric_value* is a variable, the countdown occurs, but the variable retains its initial value.

EXAMPLE:

```
REPS 3                                        # loop three times
.loop
  display "hello"
  IF REPS GOTO .loop                        # decrement; if REPS greater
  display "goodbye"                        # than 0, branch to .loop;
```

RETURN

return to a calling program

FORMAT: *RETURN numeric_constant*

RETURN returns control to the menu system or the calling BLAST script. @STATUS of the calling script is set to *numeric_constant*, or 0 if no numeric constant is specified.

IMPORTANT: When writing a script that will be CALLED, you may *not* want to assign a RETURN value of 1 if the calling script checks @STATUS to determine the success of CALL. Since @STATUS is set to 1 when

a CALLED script cannot be found, assigning a value of 1 in the RETURN statement would invalidate the check of CALL.

EXAMPLE:

```
RETURN 2      # return with @STATUS set to 2
```

SAVE

save a BLAST setup

FORMAT: *SAVE*
 SAVE saves the current setup.

EXAMPLE:

```
SAVE                        # save current setup
```

SELECT

select a system setup

FORMAT: *SELECT string_value*
 SELECT loads a setup from the directory specified by the SETUPDIR environment variable. The setup name should not include the .SU extension. This statement operates like the Offline menu Select command. If the setup has been successfully loaded, @STATUS is set to 0.

EXAMPLE:

```
SELECT "Blaster"  
If OK display "Setup successfully loaded."  
Else display "Couldn't load setup."
```

SET

set script variables to a string

FORMAT: *SET variable = string_value*
 SET assigns a value to a variable. *SET* differs from the *LET* statement in that mathematical operations cannot be performed in a *SET*. If *variable* is a reserved variable, the resulting value of @STATUS is undefined.

EXAMPLE:

```
SET @command = "blast -h"  
SET @BAUDRATE = "9600"           # set baud rate in setup  
SET @PARITY = "NONE"             # set parity in setup
```

SETTRAP

capture commport data to a script variable

FORMAT: *SETTRAP* *variable*, *numeric_constant1* [, *numeric_constant2*]

SETTRAP prepares a TTRAP command to capture incoming data into a user-defined variable. Note that SETTRAP will not perform the capture itself—one or more TTRAPs must follow. Once a SETTRAP is issued, it remains in effect until another SETTRAP is issued; therefore, one SETTRAP can be used for multiple TTRAPs.

Variable specifies the destination for the TTRAP data. It may be either a new or previously used variable.

Numeric_constant1 defines the maximum number of characters to save into the variable. It must be greater than 0 and may be up to 139 long.

Only the last incoming characters, specified by *numeric_constant1*, will be saved. When set to 0, SETTRAP is disabled completely and the TTRAP(s) following will operate normally.

Numeric_constant2 contains the maximum amount of characters the TTRAP(s) will check for a match. If this value is reached, the TTRAP(s) will return to the calling script with @STATUS set to -5, and the TTRAP internal counter will be reset. Note that this is not on a per-TTRAP basis; the value is accumulated over one or more TTRAPs. This feature may be disabled by setting *numeric_constant2* to 0 or omitting it.

EXAMPLE:

```
SETTRAP @CAP, 10, 85           # set TTRAP to capture data into  
                               # @CAP--10 chars max, TTRAP exits  
                               # if 85 chars are received before  
                               # TTRAP matches string or times out  
ttrap 6, "\015"               # trap next carriage return  
SETTRAP @CAP, 20              # No character count, so TTRAP  
                               # will timeout or match string.  
                               # 20 chars are placed in @CAP  
ttrap 45, "Logout"
```

STRCAT

combine strings

FORMAT: *STRCAT* variable, string_value [,string_value ...]

STRCAT appends *string_value* to variable.

EXAMPLE:

```
set @string1 = "abc"
set @string2 = "xyz"
STRCAT @string1, @string2    # append string2 to string1
display "alpha=", @string1   # display abcxyz
```

STRINX

find the first occurrence of one string in another

FORMAT: *STRINX* string_value1, string_value2

STRINX finds the first occurrence of *string_value2* in *string_value1*. @STATUS is set to the starting character position of *string_value2* in *string_value1*, or set to 0 if there is no match.

EXAMPLE:

```
set @string1 = "0123456"
STRINX @string1, "3"        # look for pattern "3"
display "The number 3 occurs at position ", @STATUS
```

STRLEN

determine the length of a string

FORMAT: *STRLEN* variable

STRLEN sets @STATUS to the length of *variable*.

EXAMPLE:

```
STRLEN @string
display "The length of @string is", @STATUS
```

STRTRIM

extract part of a string

FORMAT: *STRTRIM variable, numeric_value1, numeric_value2*

STRTRIM extracts a substring from *variable*. *Variable* is reset to the substring that begins at position *numeric_value1* and ends at position *numeric_value2*. If the original string will be required for further processing, a copy of it should be made before operating with STRTRIM, because STRTRIM changes the contents of *variable*. If the variable in a STRTRIM statement is a reserved variable, the behavior of STRTRIM is undefined. For this reason, we strongly recommend that you *not* use reserved variables in a STRTRIM statement.

EXAMPLE:

```
set @name = "A:\\file.dat"           # name is A:\file.dat
set @drive = @name                   # make a copy of string
STRTRIM @drive, 1, 2                 # trim out drive name
display "Filename is ", @name
display "Drive name is ", @drive?
```

TCAPTURE

enable text file capture

FORMAT: *TCAPTURE {ON [APPEND | OVERWRITE] | OFF} string_value*

TCAPTURE enables or disables text capturing while in Terminal mode. TCAPTURE ON enables Capture mode, and TCAPTURE OFF disables it. APPEND and OVERWRITE are used only with ON to indicate whether an existing file should be appended or overwritten. If neither is specified, APPEND is assumed.

@STATUS is set to 0 if *string_value* is a valid filename that can be written to; otherwise, @STATUS is set to an error code. TCAPTURE OFF does not affect @STATUS. No data is captured until one of the following is executed: TSEND, TTRAP, TUPLOAD, or WAIT with the CARRIER or IDLE option.

IMPORTANT: After issuing a TCAPTURE command, you should perform a WAIT IDLE or TTRAP to be sure that a stopping point has been reached in the data stream before exiting.

EXAMPLE:

```
TCAPTURE ON APPEND "test.cap"           # capture on; append
                                           # to file test.cap
if @STATUS not = "0"                     # if not OK
    display "can't enable capture"       # write to screen
    return 2                             # return error code
end
tsend "type bob.mail", CR                # send command to the
                                           # the remote system
wait 10 idle                             # wait till no comm
                                           # port activity
TCAPTURE OFF                             # turn capture off
```

TERMINAL

become a terminal

FORMAT: *TERMINAL*

TERMINAL puts BLAST into Terminal mode, allowing the user to interact with the remote computer. Control cannot return to the script until the user types *ATTN ATTN*. TERMINAL will not function if BLAST is started with the */b* switch (batch mode) or */n* switch (no display, unless the */n* switch setting has been reset in the session—for example, in a script with the following command: SET @SCRLREG = "ON").

EXAMPLE:

```
display "Script paused..."
TERMINAL
display "Script continuing..."
```

TSEND

send strings to the remote computer

FORMAT: *TSEND {BREAK | CR | LF | string_value},...*

TSEND sends breaks, carriage returns, line feeds, or strings to the remote computer. Any combination of strings, line terminating characters, and/or breaks can be sent. If TSEND is successful, @STATUS is set to 0.

NOTE: @STATUS does *not* reflect whether the remote computer successfully *received* a string, only that the TSEND command was successfully executed. Also note that some operating systems (including DOS) expect a CR/LF instead of a LF at the end of a line.

Take this into consideration and use CR/LF instead of LF for these systems. You might define an end-of-line variable at the beginning of a BLAST script to make these programs easily transportable to other systems.

EXAMPLE:

```
set @endline = "CR/LF"
TSEND BREAK # send break signal
TSEND "ATDT", @PHONENO, @endline # dial the modem
```

TTRAP

trap for output from the remote computer

FORMAT: *TTRAP [mm:ss | ss,] string_value1[,...string_value8]*

TTRAP pauses the BLAST script in Terminal mode, testing data flow to the communications port. When TTRAP sees one of the string values, it continues to the next statement. If mm:ss (minutes:seconds) is given and none of the string values is received in that length of time, TTRAP times out. TTRAP sets @STATUS to the number of the string that was found, or sets @STATUS to 0 if TTRAP timed out.

EXAMPLE:

```
set @x = "NO CARRIER"
TTRAP 30, "CONNECT", @x
if @STATUS = "0" write "Timeout on trap"
if @STATUS = "1" write "Connected!"
if @STATUS = "2" write "No carrier!"
```

TUPLOAD

upload a text file to the remote system

FORMAT: *TUPLOAD string_value*

TUPLOAD opens the file specified by *string_value* and sends the text to the remote computer. The transmission is paced by any flow control options specified in the setup. TUPLOAD sets @STATUS to 0 on completion of the text upload. If the upload is unsuccessful, @STATUS is set to the applicable BLAST error code. For example, if the file could not be found, @STATUS is set to 51 (error opening data file).

Some computers buffer the flow of data extensively. This means the TUPLOAD statement may complete well before all the characters clear the local and remote computer buffers. Thus, after a TUPLOAD

command has been issued, it is a good idea to TTRAP for characters signaling the end of the upload or do a WAIT mm:ss IDLE. Exiting BLAST before the buffers are emptied may cause BLAST to terminate abnormally. See “Downloading Text” on page 192.

EXAMPLE:

```
TUPLOAD "file.txt"
wait 3 idle
ttrap 30, "-End-"           # -End- is the last line if
                             # @status = "0"
    ttrap 30, "-End-"       # wait 30 more seconds
    if @STATUS = "0" return # Return an error end
wait 10 idle                 # Make sure buffer is empty
```

UPPER

convert a variable to uppercase

FORMAT: *UPPER variable*

UPPER changes all lowercase characters in variable to uppercase. When the variable in an UPPER statement is a reserved variable, the behavior of UPPER is undefined. For this reason, we strongly recommend that you *not* use reserved variables in an UPPER statement.

EXAMPLE:

```
UPPER @salesdata
```

WAIT

wait for time to pass

FORMAT: *WAIT {mm:ss | string_value}*

WAIT pauses the BLAST script for *mm* minutes and *ss* seconds. *String_value* must be in the format *mm:ss*. The maximum value is 60 minutes (60:00).

EXAMPLE:

```
WAIT 2:02           # wait two minutes, two seconds
WAIT 2              # wait two seconds
WAIT 60:00          # wait one hour
```


WAIT CARRIER

wait for a phone call

FORMAT: *WAIT {mm:ss | string_value} CARRIER*

WAIT CARRIER pauses the BLAST script *mm* minutes and *ss* seconds, or until the modem raises carrier detect. If the modem raises carrier detect, @STATUS is set to 0. If the statement times out, @STATUS is set to a nonzero value. The maximum value is 60 minutes (60:00). Carrier detection may not be available on some communications ports if the device driver does not provide the signal. Make sure that the modem and cable are configured to indicate when the carrier signal is present.

EXAMPLE:

```
WAIT 2:02 CARRIER        # wait two minutes and
                           # two seconds for a call
WAIT 12:00 CARRIER      # wait 12 minutes for a call
WAIT 12 CARRIER         # wait 12 seconds for a call
```

WAIT IDLE

wait for communications port activity to finish

FORMAT: *WAIT {mm:ss | string_value} IDLE*

WAIT IDLE pauses the script until no characters are received on the communications port for *mm* minutes and *ss* seconds. The maximum value is 60 minutes (60:00). If the script pauses for the specified time, @STATUS is set to 0; if the script does *not* pause for the specified time, @STATUS is unchanged.

EXAMPLE:

```
WAIT 2:02 IDLE            # wait for two minutes and
                           # two seconds of idle
WAIT 1:00 IDLE            # wait for one minute of idle
WAIT 1 IDLE               # wait for one second of idle
```

WAIT UNTIL

wait for a specified time of day

FORMAT: *WAIT UNTIL {hh:mm | string_value}*

WAIT UNTIL pauses the script until the time is *hh* hours (24-hour clock) and *mm* minutes.

EXAMPLE:

```
WAIT UNTIL 2:02      # wait till 2:02 am
WAIT UNTIL 1:00      # wait till 1:00 am
WAIT UNTIL 13:30     # wait until 1:30 pm
```

WERROR

write an error message to the second menu line

FORMAT: *WERROR string_constant*

WERROR writes an error message to the operator and the log file. If @ONERROR is set to the default setting, STOP, WERROR pauses for a key to be pressed before continuing. Do not use this statement when writing a BLAST script that will be unattended unless @ONERROR is set to CONTINUE.

EXAMPLE:

```
WERROR "no response"  # display error message
return 2              # return with @STATUS set to 2
```

WRITE

write a message to the second menu line

FORMAT: *WRITE string_constant*

WRITE displays a message to the operator and the log file (without pausing as in WERROR).

EXAMPLE:

```
WRITE "dialing CHICAGO"
```

Chapter 16

BLASTscript Reserved Variables

BLASTscript reserved variables are an important part of any program that tests the condition of the communication session or the results of other statements.

There are two types of BLASTscript reserved variables: read-only and read/write. BLAST scripts can test a physical signal or logical condition using read-only variables. With read/write variables, scripts may not only test but also change a condition by using the SET command.

Reserved variables that reflect multiple-choice setup fields may be SET by using a value offered by the setup field. For example,

```
SET @DCDLOSS = "ABORT"
```

will change the value of the DCD Loss Response setup parameter in the BLAST protocol to ABORT.

In the following descriptions, if the reserved variable is associated with a setup field, the setup field will be indicated by italic print as the last line of the variable description. The characteristics of such

fields are described in Chapter 5. The default value of the reserved variable is indicated by bold print and brackets.

@7BITCHN read/write
YES **[NO]**

For BLAST protocol transfers, specifies the data-path width.

BLAST Protocol subwindow: 7-Bit Channel

@ACKFREQ read/write
1 – window size **[4]**

For BLAST protocol transfers, specifies the frequency at which an acknowledgement from the receiving system is requested. The frequency is measured in number of packets sent. See also @WDWSIZ (page 274).

BLAST Protocol subwindow: Ack Request Frequency

@ANSIAUTOWRAP read/write
[YES] [NO]

For ANSI emulation, specifies automatic wrapping of lines longer than 80 characters.

ANSI Emulation subwindow: Auto Wrap

@ANSILEVEL read/write
2.x **[3.x]**

For ANSI emulation, specifies the correct level of ANSI for your system. Some applications require ANSI Level 2 . x.

ANSI Emulation subwindow: ANSI Level

@APROTO read/write
YES **[NO]**

For BLAST protocol transfers, specifies whether the BLAST “A” Protocol will be used. Set this field to YES to communicate with older versions of BLAST.

BLAST Protocol subwindow: Use “A” Protocol

@ARGnread/write
user-defined

Stores variables passed from the operating system command line. This variable is a read-only variable where *n* specifies the argument, from 0 to 9 (@ARG0, @ARG1, etc.). The command line must include a setup name before the first command line parameter is given (see “Command Line Switches” on page 10).

@ATTKEYread/write
any Control Key [**^K**]

Specifies the attention key (*ATTN*).

Setup field: Attention Key

@AUTOGROUND
(background only)read/write
[**YES**] NO BEEP

Specifies whether a BLAST script running in the background automatically switches to foreground when user input is required. The user must then manually return BLAST to the background. When this variable is set to **NO**, no automatic switching occurs. When this variable is set to **BEEP**, BLAST will give three quick beeps without swapping screens

@AUTOLFINread/write
YES [**NO**]

Specifies whether BLAST—while in Terminal mode—inserts a linefeed character after every carriage-return character displayed.

Setup field: AutoLF In

@AUTOLFOUTread/write
YES [**NO**]

Specifies whether BLAST—while in Terminal mode—inserts a linefeed character after every carriage-return character that leaves the communications port.

Setup field: AutoLF Out

@BAUDRATE

read/write
 300 600 1200 2400 4800
[9600] 19.2 38.4 57.6 115K

Specifies the speed of your PC's communications port. The default value of this variable is set during the BLAST installation process. Some systems may not support higher baud rates.

Setup field: Baud Rate

@BLASTDIR

read-only

Stores the directory path for BLAST executable files and support files. BLASTDIR is specified during initial installation, but it may be modified by setting the BLASTDIR environment variable or the BLASTDIR option in BLAST.OPT (see "Environment Variables" on page 8 and "BLAST.OPT Settings" on page 20).

@CHARDLY

read/write
[0] – 999

Specifies the time delay (in hundredths of a second) between each character sent to the remote computer when uploading text or executing TSEND commands.

Setup field: Char Delay

@COMMPORT

read/write
 any valid device

Specifies the communications port or LAN driver to use for the current session. For serial connections, acceptable values are COM1: – COM8:, BIOS1:, BIOS2:, or a user-defined label.

NOTE: For LAN connections, acceptable values are network drivers that were defined during the installation process or in a valid BLAST.OPT COMMPORT assignment. Note that this variable only stores the value of the Connection Type; the Connection Name is stored in @NETSERVICE.

See your Installation Guide for more information on installing LAN drivers and destination names.

The default value of this variable is set during the BLAST installation process.

Setup field: Connection

@COMP_LVL

read/write

0 – 6 [4]

For BLAST protocol transfers, specifies the maximum sending and receiving compression levels to be used. Level 0 specifies no compression; level 6 specifies the highest level of compression. Setting this variable is effectively equal to setting both the @RCOMP_LEV and @SCOMP_LEV reserved variables (see “Compression Levels” on page 121).

@CONNTIMO

read/write

0 – 999 [60]

Specifies the number of seconds BLAST will wait for a network connection before timing out. This field has no effect on serial connections.

Setup field: Connection T/O

@CONTIMO

read/write

0 – 999 [120]

Used with older versions of BLAST. For BLAST protocol transfers, specifies the number of seconds that BLAST will wait for a packet of data from the remote computer before timing out.

IMPORTANT: This reserved variable has been replaced by the reserved variable @INACTIMO and should not be used. Do not confuse it with the @CONNTIMO reserved variable described directly above.

@CTS

read-only

Stores the Clear-to-Send (CTS) device status. If @CTS is set to 1, the device, usually a modem, is ready to receive characters. @CTS is set to 0 if the device is not ready to receive characters. The value of this variable is valid only when BLAST is talking to a hardware port.

@D/S_BITS

read/write

7/1 7/2 [8/1] 8/2

Specifies data and stop bits for the communications port.

Setup field: Data/Stop Bits

@DATE

read-only

Contains the current date. By default the format is *mm/dd/yy*. This format may be changed using the reserved variable @DATEFORMAT or the /y switch (see the discussion of /y on page 12).

@DATEFORMAT

read/write
template

Sets the format of the @DATE variable. Setting the @DATEFORMAT reserved variable overrides the format in which BLAST was started. The format of the output of the @DATE reserved variable will be determined by the @DATEFORMAT template set by the user. The value of the replacement sequences are as follows:

- %A full weekday name (Monday)
- %a abbreviated weekday name (Mon)
- %B full month name (January)
- %b abbreviated month name (Jan)
- %c standard date/time representation (%a %b %d %H:%M:%S %Y)
- %d day-of-month (01-31)
- %H hour (24 hour clock) (00-23)
- %I hour (12 hour clock) (01-12)
- %j day-of-year (001-366)
- %M minute (00-59)
- %m month (01-12)
- %p local equivalent of AM or PM
- %S second (00-59)
- %U week-of-year, first day Sunday (00-53)
- %W week-of-year, first day Monday (00-53)
- %w weekday (0-6, Sunday is 0)
- %X standard time representation (%H:%M:%S)
- %x standard date representation (%a %b %d %Y)
- %Y year with century
- %y year without century (00-99)
- %Z time zone name
- %% percent sign

For example, to set @DATEFORMAT to generate a date in the format of 19-March-1998, your script would read

```
set @DATEFORMAT = "%d-%B-%Y"
```


@DCD

read-only

Stores the Carrier-Detect status from the modem. If @DCD is set to 1, the carrier is detected by the modem. If @DCD is set to 0, the modem does not sense a carrier from another modem. The modem must be set appropriately for this variable to reflect the state of the data carrier; and the modem cable, if present, must have the appropriate conductor.

@DCDLOSS

read/write
ABORT **[IGNORE]**

For BLAST protocol transfers, specifies whether BLAST will ABORT after or IGNORE DCD loss. This feature requires appropriate modem initialization (see discussion of @DCD above).

BLAST Protocol subwindow: DCD Loss Response

@DGCURSTYPE

read/write
[REVERSE VIDEO]
NONE UNDERLINE

For DG emulation, specifies the cursor type for DG emulation.

DG Emulation subwindow: Cursor Type

@DGDATABITS

read/write
7 **[8]**

For DG emulation, specifies data bits for DG emulation.

DG Emulation subwindow: Data Bits

@DGPRTMODE

read/write
[PRINT SCREEN]
PRINT WINDOW
PRINT PASSTHROUGH ON
SIMULPRINT ON PRINT FORM

For DG emulation, specifies local print key action.

DG Emulation subwindow: Local Print Option

@DGPRTWIND

read/write
[ENABLED] DISABLED

For DG emulation, specifies whether the local print key and keyboard shortcuts for DG local print commands are enabled.

DG Emulation subwindow: Print Window

@EFERROR

read/write

For BLAST protocol, returns the error code of the last error in a file transfer (see Appendix A). If no error occurs during the BLAST session, @EFERROR will remain set at 0. @EFERROR should be reset to 0 for continued testing during a session. Because BLAST queues filetransfer requests and then continues execution until ESC is encountered, testing @EFERROR within a FILETRANSFER-ESC block may not produce expected results.

Following completion of a BLAST protocol file transfer, @EFERROR will be set to a transfer file management error (error 31–49; see “Transfer File Management” on page 322) or one of the following values reflecting the way in which Filetransfer mode was exited:

- 0 No errors
- 1 Initialization error
- 2 Local operator ended activity with *ATTN*
- 3 Remote disconnect
- 4 Never got starting message (Logon Timeout)
- 5 Lost communications with remote system (Inactivity Timeout)
- 6 Private network error; private network version of BLAST required
- 7 DCD loss during Filetransfer logon
- 8 DCD loss during Filetransfer session

Example:

```
connect
set @protocol = "BLAST" # BLAST protocol only!!
set @EFERROR = "0"
filetransfer
send
test1.fil
recv1.fil
to
esc
if @EFERROR not = "0"
```

```

display "Error number = ", @EFERROR, "occurred"
display "See Chapter 16 and Appendix A for details."
set @EFERROR = "0"
end
disconnect
return 0

```

@EFLOG

read/write
filename

Specifies a separate error-free log file that will log all filetransfer session errors or completions, or both, depending on the setting of @EFLOGGING. The default of @EFLOGGING is BOTH. Setting @EFLOG = "" (null) turns off filetransfer session logging. The information written to the file appears exactly as it does on the user's screen, allowing easier parsing of a filetransfer session.

@EFLOGGING

read/write
[BOTH] ERRORS
COMPLETIONS

Specifies whether the log file named in @EFLOG will log filetransfer ERRORS, COMPLETIONS, or BOTH. Refer to @EFLOG above for further information.

@ELAPTIME

read-only

Contains the current elapsed online time for a BLAST communications session. The value is in *hh:mm:ss* format. This variable can be reset within a BLAST script by any SET statement, for example:

```
set @ELAPTIME = "it doesn't matter"
```

The current value is not checked and is simply reset to 00:00:00.

@EMULATE

read/write
any valid terminal emulator

Specifies the terminal type to emulate in Terminal mode. Acceptable values are VT320, VT220, VT100, VT52, ANSI, D461, D411, D410, D200, TV920, D80, ADM3A, WYSE60, WYSE50, HP2392, IBM3101, and TTY.

Setup field: Emulation

@ENABLEFSread/write
YES [NO]

For BLAST protocol transfers, enables the /FWD and /STR file transfer switches, which automatically delete files.

BLAST Protocol subwindow: Enable /FWD and /STR

@ENABLERCMDread/write
[YES] NO

For BLAST protocol transfers, enables the /OVW (overwrite) file transfer switch and allows system commands to be sent from the remote system.

BLAST Protocol subwindow: Enable /OVW and Remote Cmds

@FILTERread/write
ON [OFF]

For BLAST protocol transfers, specifies whether the protocol filter is turned on. When @FILTER is set to ON, BLAST strips VT sequences sent from a mainframe protocol converter, preventing BLAST protocol from labeling these as bad blocks.

BLAST Protocol subwindow: Filtering

@FULLSCRread/write
[YES] NO

Specifies whether the top four lines of the BLAST menu region will be suppressed while in Terminal mode. Set to YES to suppress the menu and NO to enable it.

Setup field: Full Screen

@GROUND
(background only)read/write
FOREGROUND
[BACKGROUND]

Specifies whether the PC will display the foreground application screen or the background BLAST screen. @STATUS is set to 0 if the change in ground was successful.

@HOTKEY read/write
(background only) **[ENABLED]** DISABLED

Enables/disables the background Hot Key (ALT H). When disabled, the Hot Key has no effect on background/foreground switching; when enabled, the Hot Key toggles between the foreground application and BLAST in background mode.

IMPORTANT: The user will not be able to return to BLAST running in the background if the foreground is displayed and the Hot Key is disabled. If a script disables @HOTKEY without re-enabling it or setting @GROUND to background, then the BLASTAT.EXE program will be required to terminate BLAST (see the discussion of BLAST.EXE on page 15).

@HPBLKTERM read/write
any control character [**^^**]

For HP emulation, specifies the block terminator character. A block terminator character is transmitted to the remote computer at the end of a transfer operation.

HP Emulation subwindow: BlkTerminator

@HPDESTRBS read/write
[NO] YES

For HP emulation, specifies that BACKSPACE erase the character above the cursor after moving the cursor one character to the left.

HP Emulation subwindow: Destructive BS

@HPFLDSEP read/write
any control character [**^**]

For HP emulation, specifies the field separator character. When you press ENTER while in block page mode containing a formatted display, a field separator character is automatically transmitted at the end of each protected field (except the final one).

HP Emulation subwindow: FldSeparator

@HPINHDC2

read/write
[NO] YES

For HP emulation, specifies inhibition of D1/D2/D1 handshaking when data is transferred from the emulator to the remote computer.

@HPINHDC2 is used with @HPINHHNDSHK to determine what type of handshaking is used. If @HPINHDC2 is set to YES and @HPINHHNDSHK is set to NO, D1 handshaking is used. If both @HPINHHNDSHK and @HPINHDC2 are set to YES, all handshaking is inhibited.

HP Emulation subwindow: Inh DC2(H)

@HPINHHNDSHK

read/write
[NO] YES

For HP emulation, specifies inhibition of D1 handshaking when data is transferred from the emulator to the remote computer.

@HPINHHNDSHK is used with @HPINHDC2 to determine what type of handshaking is used. If @HPINHHNDSHK is set to YES and @HPINHDC2 is set to NO, D1/D2/D1 handshaking is used. If both @HPINHHNDSHK and @HPINHDC2 are set to YES, all handshaking is inhibited.

HP Emulation subwindow: InhHndShk(G)

@HPINHWRAP

read/write
[NO] YES

For HP emulation, specifies inhibition of automatic text wrapping. If @HPINHWRAP is set to NO, text automatically wraps; if it is set to YES, when the cursor reaches the right margin, it remains there (with succeeding characters overwriting the existing character) until a carriage return or other cursor movement is performed.

HP Emulation subwindow: InhEolWrp(C)

@HPLINEPAGE

read/write
[LINE] PAGE

For HP emulation, specifies whether a line or a page at a time is transmitted when operating in block mode.

HP Emulation subwindow: Line/Page(D)

@HPSTARTCOL

read/write
[0] YES

For HP emulation, specifies the position of the left margin if no logical start-of-text pointer has been generated.

HP Emulation subwindow: Start Col

@HPTERMID

read/write
[2392A] any valid terminal ID

For HP emulation, specifies what terminal identification will be sent to the remote computer when a Terminal Id request (ESC *S^) is made.

HP Emulation subwindow: Terminal Id

@INACTIMO

read/write
0 – 999 **[120]**

For BLAST protocol transfers, specifies the number of seconds that BLAST will wait for a packet of data from the remote computer during BLAST transfer, before timing out.

NOTE: This variable replaces the @CONTIMO variable of older versions.

BLAST Protocol subwindow: Inactivity T/O

@KBCHECK

read/write
1 – 3 **[2]**

For Kermit transfers, specifies the level of error-detection.

Kermit Protocol subwindow: Block-Check-Type

@KDELAYOS

read/write
1 – 99 **[5]**

For Kermit transfers, specifies the number of seconds of delay between the recognition of a Send command and the actual beginning of the transmission.

Kermit Protocol subwindow: Delay

@KEYBOARD

read/write
[ON] OFF

Specifies whether data may be entered from the keyboard. If **ON**, the keyboard is unlocked and may be used. If **OFF**, the keyboard is locked. When started in video-suppress mode (/n command line switch—see page 11), **BLAST** sets this variable to **OFF**. If **@KEYBOARD** is set to **ON**, it returns the value 1; if it is set to **OFF**, it returns the value 0.

IMPORTANT: If you need to enter input from the keyboard after running a script with **@KEYBOARD** set to **OFF**, remember to reset **@KEYBOARD** to **ON** in your script. If you do not and include a **RETURN** statement in your script, you will be returned to the Online menu, but your keyboard will be locked.

@KEYFILE

read/write
filename

Specifies a user-defined keyboard map for a particular keyboard or application. Keyboard maps are created with **BLASTKBD**, the **BLAST** keyboard remapping utility (see “Keyboard Mapping Utility” on page 296 for details).

Setup field: Keyboard File

@KFILETYP

read-only
[BINARY]

For Kermit transfers, specifies the type of file being transferred. **BINARY** is the only possible value.

Kermit Protocol subwindow: Transfer Type

@KFNAMCONV

read/write
[YES] NO

For Kermit transfers, converts a filename from local format to common format.

Kermit Protocol subwindow: Filename Conversion

@KREOPKTread/write
^A – ^Z [**^M**]

For Kermit transfers, specifies a control character to terminate each packet received. The same control character must also be used by the remote Kermit.

Kermit Protocol subwindow: End-of-Packet Char

@KRETRYread/write
1 – 99 [**10**]

For Kermit transfers, specifies the number of times Kermit will attempt to send a single packet before aborting.

Kermit Protocol subwindow: Retry Limit

@KRPADCHread/write
^A – ^Z [**^@**]

For Kermit transfers, specifies an alternate character to pad each packet received by your PC.

Kermit Protocol subwindow: Pad Character

@KRPADDNGread/write
[**0**] – 99

For Kermit transfers, specifies the number of padding characters to request per packet.

Kermit Protocol subwindow: Padding

@KRPKTLENread/write
10 – 2000 [**90**]

For Kermit transfers, specifies the packet size your PC will use when it receives a file. Note that the remote Kermit's Send packet size should also be set to this length.

Kermit Protocol subwindow: Packet Size

@KRSOPKTread/write
[**^A**] – **^Z**

For Kermit transfers, specifies the control character that marks the start of each packet received by your PC. The same control character must also be used by the remote Kermit.

Kermit Protocol subwindow: Start-of-Packet Char

@KRTIMEOUTread/write
0 – 99 [**10**]

For Kermit transfers, specifies the number of seconds that your PC will wait to receive a packet before requesting that it be resent.

Kermit Protocol subwindow: Timeout

@KSAVEINCread/write
[**DISCARD**] **KEEP**

For Kermit transfers, specifies whether to **KEEP** or **DISCARD** files not completely received, such as a file being transferred when you abort a **Get** command.

Kermit Protocol subwindow: Incomplete File

@KSEOPKTread/write
^A – **^Z** [**^M**]

For Kermit transfers, specifies a control character to terminate each packet sent by your system. The same control character must also be used by the remote Kermit.

Kermit Protocol subwindow: End-of-Packet Char

@KSPADCHread/write
^A – **^Z** [**^@**]

For Kermit transfers, specifies an alternate character to pad each packet sent by your PC.

Kermit Protocol subwindow: Pad Character

@KSPADDNGread/write
[0] – 99

For Kermit transfers, specifies the number of padding characters to send per packet.

Kermit Protocol subwindow: Padding

@KSPKTLENread/write
10 – 2000 **[90]**

For Kermit transfers, specifies the packet size your PC will use when it sends a file. Note that the packet size of the remote Kermit must also be set to this length.

Kermit Protocol subwindow: Packet Size

@KSSOPKTread/write
[^A] – ^Z

For Kermit transfers, specifies the control character that marks the start of each packet sent by your PC. The same control character must also be used by the remote Kermit.

Kermit Protocol subwindow: Start-of-Packet Char

@KSTIMEOUTread/write
0 – 99 **[10]**

For Kermit transfers, specifies the number of seconds that your PC will wait for the acknowledgement of a packet before resending it.

Kermit Protocol subwindow: Timeout

@KWARNINGread/write
[ON] OFF

For Kermit transfers, specifies whether Kermit will automatically rename a received file if another file with the same name already exists in the current directory. If @KWARNING set to ON, Kermit automatically renames the file by appending a number (0001, 0002, etc.) to the filename; if it is set to OFF, Kermit overwrites the file.

Kermit Protocol subwindow: Warning

@LAUNCHSTread/write
any ASCII string [**\r**]

For BLAST protocol transfers, specifies the launch string to be appended to BLAST protocol packets. Any ASCII string may be used, with control characters represented by a backslash followed by a three-digit octal number (see the discussion of special control characters on page 202). The default is a carriage return (**\r**). This variable may be necessary for protocol converter connections.

BLAST Protocol subwindow: Launch String

@LINEDLYread/write
[0] – 999

Specifies the length of time (in tenths of a second) that BLAST pauses after sending a line of characters and a carriage return during a text upload.

Setup field: Line Delay

@LOECHOread/write
YES [**NO**]

Specifies whether BLAST will echo typed characters to the screen while in Terminal mode. If **@LOECHO** is set to YES, BLAST will display typed characters before sending them out the communication port; if **@LOECHO** is set to NO, the characters will be displayed only if the remote computer sends them back.

If **@LOECHO** is set to YES and double characters are displayed on the screen, change the setting to NO.

Setup field: Local Echo

@LOGDATEFORMATread/write
template

Specifies the format of the date written in the date stamp of the log file. Setting **@LOGDATEFORMAT** overrides the two-digit year or four-digit year mode in which BLAST was started. The format of dates written in the log file will be determined by the template set by the user. The value of the replacement sequences are the same as those described above in the **@DATEFORMAT** reserved variable.

@LOGFILE

read/write
filename

Specifies the name of the log file that will record all communications session activity. Setting @LOGFILE = @LOGFILE flushes the log file buffers to disk. Setting @LOGFILE = " " closes the current log file.

Setup field: Log File

@LOGTIMEFORMAT

read/write
template

Sets the format of the time written in the time stamp of the log file. The format of times written in the log file will be determined by @LOGTIMEFORMAT template set by the user. The value of the replacement sequences are the same as those described above in the @DATEFORMAT reserved variable.

@LOGTIMO

read/write
0 – 999 [120]

For BLAST protocol, specifies the number of seconds that BLAST will attempt to establish a filetransfer session with the remote computer before aborting. @LOGTIMO affects BLAST protocol File-transfer and Access modes. If @LOGTIMO is set to 0, no timeout will occur and BLAST will attempt to establish a filetransfer session with the remote computer indefinitely.

BLAST Protocol subwindow: Logon T/O

@MAXMEM

(background only)

read/write
user-defined

Stores the amount of memory used from the BLAST.OPT mempool during the current BLAST script execution. This is not the amount of memory BLAST is presently using, but the maximum that BLAST has used so far. This read-only variable can help in minimizing the amount of memory BLAST uses during background operation.

@MODEM

read/write
any valid modem type

Specifies the modem type on the local computer. The name must be defined in the MODEMS.SCR library or exist as a separate script.

Setup field: Modem Type

@NETSERVICE

read/write
any valid commport
or network driver

Specifies the Connection Name for the LAN driver defined in @COMMPORT. See your Installation Guide for more information on installing LAN drivers and destination names. The default value of this variable is set during the installation process if a LAN driver was installed.

@NUMDISC

read/write
0 – 9 **[3]**

For BLAST protocol, sets the number of additional disconnect blocks (after the first disconnect block) that BLAST sends when exiting Filetransfer mode. Possible values are 0–9. The default value of 3 indicates four total disconnect blocks.

BLAST Protocol subwindow: Number of Disconnect Blocks

@ONERROR

read/write
[STOP] CONTINUE

Specifies BLAST's response to nonfatal BLASTscript errors. A nonfatal error is one that results in the message "Press any key to continue."

When @ONERROR is set to STOP, BLAST will pause when an error is encountered, display the appropriate message, and wait for the user to press a key before continuing. When @ONERROR is set to CONTINUE, BLAST will display the same message, pause for one second, and then automatically continue script execution.

@ORGANS

read/write
[ORIGINATE] ANSWER

Specifies how the Connect command will operate. If @ORGANS is set to ANSWER, Connect will wait for a remote computer to establish the communications link. If it set to ORIGINATE, Connect will try to dial a number.

NOTE: For node-to-node connections, one system must be set to ORIGINATE and the other system set to ANSWER.

Setup field: Originate/Answer

@PAKTSZ

read/write
1 – 4085 **[256]**

For BLAST protocol transfers, specifies the size of the packet. For LAN operations, it also indicates the size of the packets delivered to the network when using the NetBIOS and IPX drivers for node-to-node file transfers.

Setup field: Packet Size

@PARITY

read/write
[NONE] EVEN ODD
MARK SPACE

Sets the parity of the communications port. This setting should match that of the remote system.

Setup field: Parity

@PASSWORD

write only
user-defined

Stores the user's password for the remote computer. The SYSTEMS.SCR library program uses @PASSWORD to answer prompts from a multi-user computer. The CONNECT command will prompt the user to enter a password if none is specified in the Setup. Thereafter, the variable @PASSWORD contains the value entered by the user. For security, the value of @PASSWORD cannot be displayed to the screen. This feature applies to all string values that match @PASSWORD. Thus, script commands such as

```
set @trick = @PASSWORD
display @trick
```

will *not* display the value of the password.

BLAST makes an effort to keep stored passwords secure. Unfortunately, it is a very simple task to echo a stored password off either a modem or a remote system that has echo enabled. A script as simple as “`t send @password`” can compromise stored passwords. If the security of a password is vital, BLAST recommends not storing it in the setup.

Setup field: Password

@PHONENO read/write
user-defined

Specifies the phone number of the remote computer. The `CONNECT` statement uses this number to dial out.

Setup field: Phone Number

@PROMPTCH read/write
[NONE] any ASCII character

Specifies the prompt character used during text uploads to half-duplex systems. BLAST waits after each line for the remote computer to send the prompt before sending the next line.

Setup field: Prompt Char

@PROTOCOL read/write
[BLAST] FTP KERMIT
XMODEM XMODEM1K
YMODEM YMODEM G ZMODEM

Specifies the protocol for a communications session.

Setup field: Protocol

@PULSEDIAL read/write
YES [NO]

Specifies whether to use pulse dialing. If this field is set to `YES`, pulse dialing is used; if it is set to `NO`, tone dialing is used.

BLASTscript variable: Pulse Dialing

@RBTOT

read-only

If Extended Logging is enabled, holds the total number of bytes received during the file transfer session. You must write a display statement (e.g. `Display "@RBTOT is ", @RBTOT`) for this variable to be displayed in the Extended Log file. See @XLOG (page 278) for more information.

@RBYTES

read-only

In the BLAST Extended Log, holds the number of bytes received in the current transfer. Note that this value can be different than the actual file size. You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@RCOMP_LEV

read/write
0 – 6 [4]

For BLAST protocol transfers, specifies the maximum receiving level of compression that can be used during a session. Level 0 specifies no compression; level 6 specifies the highest compression level (see “Compression Levels” on page 121).

BLAST Protocol subwindow: Receive Compression Level

@RETRAN

read/write
0 – 9999 [4]

For BLAST protocol transfers, sets the maximum number of seconds BLAST will pause before resending a packet. For example, if @WDWSIZ is set to 5 and @RETRAN is set to 30, BLAST will attempt to resend the fifth packet every thirty seconds if no acknowledgement is received.

BLAST Protocol subwindow: Retransmit Timer

@RFAILURE

read-only

For BLAST protocol, stores the number of files unsuccessfully received during a file transfer session.

@RLINEQ

read-only

For BLAST protocol transfers, stores the current receiving line quality. Possible values are GOOD, FAIR, POOR, or DEAD.

@RLQ

read-only

In the BLAST Extended Log, holds the line quality for the file being received. You must have Extended Logging enabled for this variable to return a value. Possible values are GOOD, FAIR, POOR, or DEAD. See @XLOG (page 278) for more information.

@RNAME

read-only

In the BLAST Extended Log, holds the name of the file being received. You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@ROPTIONS

read-only

In the BLAST Extended Log, holds the value of the options for the file being received. You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@RPACK

read-only

In the BLAST Extended Log, holds the number of packets received in the transfer. You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@RPTOT

read-only

In the BLAST Extended Log, holds the total number of packets received during the file transfer session. You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@RRET

read-only

In the BLAST Extended Log, holds the number of retries for the file being received. You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@RSIZE

read-only

In the BLAST Extended Log, holds the size of the file being received. You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@RSTART

read-only

In the BLAST Extended Log, holds the interrupt start point for an interrupted received file. You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@RSTATUS

read-only

In the BLAST Extended Log, holds the completion status of the file being received. Possible values are:

RCOMP – Receive completed.

LERROR – Receive not complete due to local error.

RERROR – Receive not completed due to remote error.

RINTR – Receive not completed due to operator interruption.

You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@RSUCCESS

read-only

For BLAST protocol, stores the number of files successfully received during a file transfer session.

@RTIME

read-only

In the BLAST Extended Log, holds the elapsed time for the file being received. You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@RTSCTS

read/write
[YES] NO

Specifies whether hardware flow control is enabled. Not all computers support RTS/CTS flow control. The value of this variable is valid only when BLAST is talking to a hardware port.

Setup field: RTS/CTS Pacing

@SBTOT

read-only

If Extended Logging is enabled, holds the total number of bytes sent during the file transfer session. You must write a display statement (e.g. `Display "@SBTOT is ", @SBTOT`) for this variable to be displayed in the Extended Log file. See @XLOG (page 278) for more information.

@SBYTES

read-only

In the BLAST Extended Log, holds the number of bytes sent in the current transfer. Note that this value can be different than the actual file size. You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@SCOMP_LEV

read/write
0 – 6 **[4]**

For BLAST protocol transfers, specifies the maximum sending compression level that can be used during a session. Level 0 specifies no compression; level 6 specifies the highest compression level (see “Compression Levels” on page 121).

BLAST Protocol subwindow: Send Compression Level

@SCRFILE

read/write
filename

Specifies the name of a BLAST script that will start immediately after BLAST begins execution.

Setup field: Script File

@SCRIPTERR

read/write
any integer

Returns the numeric value of the last error that occurred in the BLAST script.

@SCRLREG

read/write
[ON] OFF

Controls data display in the scrolling region (lines 5–24). If @SCRLREG is set to ON, characters received in Terminal mode will be displayed and BLAST scripts can use the DISPLAY statement. If BLAST is started in video-suppress mode (/n switch on the operating system command line—see page 11), @SCRLREG is set to OFF.

NOTE: If @SCRLREG is set to ON, it returns the value 1; if it is set to OFF, it returns the value 0.

@SETUPDIR

read-only

Contains the directory path in which BLAST setup files are stored. SETUPDIR is specified during installation, but it may be modified in the DOS environment or in BLAST.OPT. For additional information about environment variables and BLAST.OPT, see “Environment Variables” on page 8 and “BLAST.OPT Settings” on page 20.

@SFAILURE

read-only

For BLAST protocol, stores the number of files unsuccessfully sent during a file transfer session.

@SLINEQ

read-only

For BLAST protocol, stores the current sending line quality during a file transfer. Increase packet size to take advantage of clean lines, or decrease packet size to avoid problems with noisy lines. Possible values are GOOD, FAIR, POOR, or DEAD.

@SLQ

read-only

In the BLAST Extended Log, holds the line quality for the file being sent. You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@SNAME read-only

In the BLAST Extended Log, holds the name of the file being sent. You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@SOPTIONS read-only

In the BLAST Extended Log, holds the value of the options for the file being sent. You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@SPACK read-only

In the BLAST Extended Log, holds the number of packets sent in the transfer. You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@SPTOT read-only

In the BLAST Extended Log, holds the total number of packets sent during the file transfer session. You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@SRET read-only

In the BLAST Extended Log, holds the number of retries for the file being sent. You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@SRTOT read-only

In the BLAST Extended Log, holds the total number of retries for files being sent during the file transfer session. You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@SSIZE read-only

In the BLAST Extended Log, holds the size of the file being sent. You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@SSTART

read-only

In the BLAST Extended Log, holds the interrupt start point for an interrupted sent file. You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@SSTATUS

read-only

In the BLAST Extended Log, holds the completion status of the file being sent. Possible values are:

SCOMP – Send completed.

LERROR – Send not completed due to local error.

RERROR – Send not completed due to remote error.

SINTR – Send not completed due to operator interruption.

You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@SSUCCESS

read-only

For BLAST protocol, stores the number of files successfully sent during a file transfer session.

@STATUS

read/write
command-specific

Returns a condition code set by the last statement that reported a completion status. Most statements that succeed set @STATUS to 0 and return a nonzero value for an error. For example, the FILETRANSFER command sets @STATUS to 0 if Filetransfer mode was successfully entered.

Checking @STATUS at the end of a FILETRANSFER block, however, does *not* reflect the success of an entire FILETRANSFER block, but rather the @STATUS setting of the last command in the block capable of setting @STATUS. (To check the overall success of a FILETRANSFER block, use the reserved variable @EFERROR).

Some commands that return numeric results (e.g., STRINX, TTRAP) set @STATUS to 0 to indicate a null condition.

On returning from a called script, @STATUS is set to the numeric constant given in the RETURN statement, or to 0 if no numeric constant is given.

For a list of commands that set @STATUS, see “Commands That Set @STATUS” on page 204.

@STATUSLN read/write
[ON] OFF

Specifies whether the BLAST status line (line 25) is displayed. This variable is set to OFF when BLAST is started in video suppress mode (/n on the operating system command line).

@STIME read-only

In the BLAST Extended Log, holds the elapsed time for the file being sent. You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@SYSDISC read/write
user-defined

Specifies a user-defined description of the remote computer. This field may be up to 40 characters. No special processing is done based on the information in this field.

Setup field: Description

@SYSTYPE read/write
any valid system type

Specifies the remote computer type (UNIX, VMS, etc.). The SYSTEMS.SCR library uses this variable to determine how to perform certain system functions, such as logging on and disconnecting from remote multi-user computers.

Setup field: System Type

@TIME read-only

Contains the current time in *hh:mm:ss* format. This is a read-only variable; an error message will be displayed if a script attempts to write to it.

@TIMEFORMAT

read/write
template

Sets the format of the @TIME variable. Setting the @TIMEFORMAT reserved variable overrides the format in which BLAST was started. The format of the output of the @TIME reserved variable will be determined by the template set by the user. The value of the replacement sequences are the same as those described above in the @DATEFORMAT reserved variable.

@TRANSTAT

read/write
[ON] OFF

Specifies whether the File Transfer Status Area will be displayed. If @TRANSTAT is set to ON, the area is active. This variable is set to OFF when BLAST is started in video-suppress mode (/n on the operating system command line—see page 11).

NOTE: If @TRANSTAT is set to ON, it returns the value 1; if it is set to OFF, it returns the value 0.

@TRPASSWD

write-only
up to 8 characters

For BLAST protocol, specifies a password that a remote user must send before a file transfer is allowed. If this variable is set to other than null, then the remote computer must send the password before a file can be transferred to or from your computer.

NOTE: @TRPASSWD is intended to validate remote users logging onto your system. If the BLAST running on the local system executes a script that sets @TRPASSWD to something other than a **null**, the local computer will not be able to receive files without the remote computer sending the password.

BLAST Protocol subwindow: Transfer Password

@TTIME

read-only

In the BLAST Extended Log, holds the total elapsed time of the file transfer session. You must have Extended Logging enabled for this variable to return a value. See @XLOG (page 278) for more information.

@USERIDread/write
user-defined

Stores the user's identification for the remote computer. The SYSTEMS.SCR library uses this variable in answering the logon prompts from a multi-user computer.

Setup field: Userid

@USERIFread/write
[ON] OFF

Specifies whether the menu region (lines 1–4) is displayed. If @USERIF is set to ON, the menu region is displayed; if it is set to OFF, lines 1–4 become part of the scrolling region. When BLAST is started in the video-suppress mode (/n on the operating system command line—see page 11), this variable is turned OFF.

NOTE: If @USERIF is set to ON, it returns the value 1; if it is set to OFF, it returns the value 0.

@VERSION

read-only

Stores the version of BLAST that is running.

@VT8BITread/write
[7] 8

For VT emulation, specifies whether C1 control characters are represented in the 8-bit environment or as 7-bit escape sequences.

VT Emulation subwindows: 7/8 Bit Controls

@VTANSBACKread/write
user-defined ASCII string

For VT emulation, contains a message to be sent to the remote computer upon receiving an inquiry (^E). The field can be up to 30 characters in length. The default value is an empty string—nothing is sent.

VT Emulation subwindows: Answerback Msg

@VTAUTOWRAP

read/write
YES [NO]

For VT emulation, specifies whether text typed at the right margin will automatically wrap to the next line.

VT Emulation subwindows: Auto Wrap

@VTCLRSCRN

read/write
YES [NO]

For VT emulation, specifies compressed mode for video when @VTDISP132 is set to 132 or the host sends a sequence to the emulator to use 132 columns. Setting @VTCLRSCRN to YES clears the display; the value is then reset to NO.

VT Emulation subwindows: Clear Screen

@VTCOMPRESSED

YES [NO]

For VT emulation, specifies whether the monitor is in compressed mode when @VTDISP132 is set to 132 or when the host sends a sequence to the emulator to use 132 columns.

VT Emulation subwindows: 132 Compressed

@VTCURSOR

read/write
[NORMAL] APPLICATION

For VT emulation, specifies whether the cursor keys will control cursor movement or send application control functions.

VT Emulation subwindows: Cursor Keys Mode

@VTCURSTYPE

read/write
BLOCK [LINE]

For VT emulation, specifies whether the cursor is displayed as a reverse-video block or as an underline character.

VT Emulation subwindows: Cursor Type

@VTDISP132

read/write
[80] 132

For VT emulation, specifies column display for text.

VT Emulation subwindows: 80/132 Columns

@VTHSCROLL

[JUMP] SMOOTH NONE

For VT emulation, specifies how to scroll data on an 80-column display when @VTDISP132 is set to 132. SMOOTH scroll will change the view of the display only as necessary to display the cursor position. JUMP scroll will adjust the view based on the setting of @VTHSCROLLN. When NONE is selected, the display will not scroll and the cursor may disappear from view. When @VTCOMPRESSED is set to YES, the setting of @VTHSCROLL is ignored.

VT Emulation subwindows: Horizontal Scroll

@VTHSCROLLN

1 – 53 [10]

For VT emulation, specifies the number of columns to scroll when the cursor reaches the edge of the screen and @VTHSCROLL is set to JUMP.

VT Emulation subwindows: Jump Scroll Inc

@VTINTL

**[USASCII] UK
FRENCH GERMAN
ITALIAN SPANISH DANISH**

For VT emulation, specifies whether 7- or 8-bit data is used for international support. The default value, USASCII, allows 8-bit data with the high-order data used for international characters.

VT Emulation subwindows: Intl Char Set

@VTKEYPAD

[NUMERIC] APPLICATION

For VT emulation, specifies whether the numeric keypad keys will send numbers or programming functions defined by the application.

VT Emulation subwindows: Keypad Mode

@VTNEWLINE

read/write
YES [NO]

For VT emulation, specifies whether the ENTER key will move the cursor to a new line. Possible choices are NO (the ENTER key sends only a carriage return) and YES (both a carriage return and line feed are sent).

VT Emulation subwindows: New Line

@VTPRINT

read/write
[NORMAL] AUTO
CONTROLLER

For VT emulation, specifies when information is sent to the printer. In AUTO print mode, each line of received text is displayed and printed; in CONTROLLER mode, all received data is sent directly to the printer without displaying it on the screen; in NORMAL mode, the user initiates printing from the keyboard.

VT Emulation subwindows: Print Mode

@VTPRINTPAGE

read/write
[SCROLL REGION]
FULL PAGE

For VT emulation, specifies how much of the screen to print when you press the PRINT SCREEN key.

VT Emulation subwindows: Print Screen

@VTRESET

read/write
YES [NO]

For VT emulation, specifies resetting many of the VT320 operating features, such as scrolling regions and character attributes, to their factory default values upon entering Terminal mode. If @VTRESET is set to YES, the features are reset; the value is then automatically reset to NO.

VT Emulation subwindows: Reset Terminal

@VTSTATUSLN**[NONE]** INDICATOR
HOST WRITABLE

For VT320 emulation, specifies the status line to be displayed at the bottom of the screen. Setting @VTSTATUSLN to INDICATOR displays a status line showing cursor position, printer status, and modem status information; setting it to HOST WRITABLE displays a status line showing messages from the Host computer; and setting it to NONE displays the BLAST status line.

VT Emulation subwindows: Status Line

@VTTEXTCURSread/write
[YES] NO

For VT emulation, specifies whether to display the text cursor.

VT Emulation subwindows: Text Cursor

@VTUSERCHARread/write
[DEC SUPPLEMENTAL]
ISO LATIN-1

For VT320 emulation, specifies either DEC SUPPLEMENTAL or ISO LATIN-1 character set as the user preferred character set.

VT Emulation subwindows: User Pref Char Set

@VTUSERKEYSread/write
[UNLOCKED] LOCKED

For VT emulation, specifies whether the host system can change user-defined key definitions.

VT Emulation subwindows: User Def Keys.

@WDWSIZread/write
1 – **[16]**

For BLAST protocol, specifies the window size of the “B” protocol. “Window” refers to the number of BLAST protocol packets that can be sent to the remote without BLAST waiting for an acknowledgment from the remote. As packets are acknowledged, the start point of the window is adjusted, or “slides.” See “BLAST Protocol Design” on page 103 for a fuller discussion of window size.

BLAST Protocol subwindow: Window Size

@WT4ECHO

read/write
YES **[NO]**

Specifies whether BLAST will wait for the remote computer to echo each character of uploaded text before sending the next character.

Setup field: Wait For Echo

@WY25LINE

[BLAST STATUS LINE]
25th DATA LINE
STANDARD STATUS LINE
EXTENDED STATUS LINE

For WYSE emulation, specifies how the 25th line of your PC's screen will be used.

WYSE Emulation subwindow: 25th line

@WYANSBACK

read/write
user-defined

For WYSE emulation, contains a user-created message to be sent to the host when an inquiry is received.

WYSE Emulation subwindow: Answerback

@WYAUTOPAGE

read/write
YES **[NO]**

For WYSE emulation, specifies whether the cursor can move off the current page when an attempt is made to move the cursor before the home position or beyond the end of the page.

WYSE Emulation subwindow: Auto Page

@WYAUTOSCROLL

read/write
[YES] NO

For WYSE emulation, specifies whether to scroll the terminal display when the cursor reaches the bottom of a page.

WYSE Emulation subwindow: Auto Scroll

@WYAUTOWRAPread/write
[YES] NO

For WYSE emulation, specifies automatic line wrapping.

WYSE Emulation subwindow: Auto Wrap

@WYBLOCKENDread/write
[US/CR] CRLF/ETX

For WYSE emulation, specifies which characters are used to mark the end-of-line and end-of-block when the terminal is in block mode.

WYSE Emulation subwindow: Block End

@WYCOMMmode**[CHARACTER]** BLOCK

For WYSE emulation, specifies whether data is sent after each key-stroke (CHARACTER mode) or grouped in blocks (BLOCK mode).

WYSE Emulation subwindow: Comm Mode

@WYCOMPRESSEDYES **[NO]**

For WYSE emulation, specifies whether compressed mode is used when @WYDISP80 is set to 132 or the host sends a sequence to the emulator to use 132 columns. To use this feature, your PC must be equipped with an EGA or VGA adapter card with 132 column support and a compatible monitor.

WYSE Emulation subwindow: 132 Compressed

@WYCURSTYPE**[BLOCK]** LINE

For WYSE emulation, specifies whether the cursor is displayed as a reverse-video block or an underline character.

WYSE Emulation subwindow: Cursor Type

@WYDISP80read/write
[80] 132

For WYSE emulation, specifies a display of 80 or 132 columns per row.

WYSE Emulation subwindow: Columns

@WYDSPCURSOR

read/write
[YES] NO

For WYSE emulation, specifies that the cursor is visible.

WYSE Emulation subwindow: Display Cursor

@WYENTER

read/write
[CR] CRLF TAB

For WYSE emulation, specifies the character to send when the key-pad ENTER key is pressed.

WYSE Emulation subwindow: Enter

@WYEXPNDMEM

read/write
YES **[NO]**

For WYSE emulation, specifies whether expanded memory is used. Note that this is *not* related to DOS expanded memory.

WYSE Emulation subwindow: Expanded Memory

@WYPAGELEN

read/write
[1*DATA LINES]
2*DATA LINES 4*DATA LINES

For WYSE emulation, specifies the length of a screen page.

WYSE Emulation subwindow: Page Length

@WYPERSONALITY

[WYSE60]
PC Term

For WYSE60 emulation, specifies WYSE60 or PC Term personality.

WYSE Emulation subwindow: Return

@WYRETURN

read/write
[CR] CRLF TAB

For WYSE emulation, specifies the character to send when the RETURN key is pressed.

WYSE Emulation subwindow: Return

@WYSCROLLINC

1 – 53 [10]

For WYSE emulation, specifies the number of columns to scroll when the cursor reaches the edge of the screen and @WYDISP80 is set to 132 and @WYCOMPRESSED is set to NO.

Wyse Emulation subwindow: Horiz Scroll Inc

@WYSEWORD

read/write
YES [NO]

For WYSE emulation, specifies whether keys send Wordstar™ functions instead of standard key codes. The only keys affected are WYSE keys that can be remapped with the BLASTKBD utility (see “Keyboard Mapping Utility” on page 296).

WYSE Emulation subwindow: Wyseword

@WYWRITEPROT

read/write
[DIM] REVERSE NORMAL

For WYSE emulation, specifies the attribute used to display protected fields.

WYSE Emulation subwindow: Write Protect

@XLOG

read/write
ON [OFF]

Specifies whether Extended Logging is enabled. Extended Logging provides detailed information about BLAST protocol file transfers. The following variables return values during Extended Logging:

@SNAME	@RNAME	@STIME	@RTIME
@SOPTIONS	@ROPTIONS	@SPACK	@RPACK
@SSTATUS	@RSTATUS	@SRET	@RRET
@SSIZE	@RSIZE	@SPTOT	@RPTOT
@SSTART	@RSTART	@SRTOT	@RRTOT
@SBYTES	@RBYTES	@SBTOT	@RBTOT
@SLQ	@RLQ	@TTIME	

During Extended Logging, all the values listed above are written to the log file except for @RBTOT and @SBTOT, which may be written to the log file by issuing a display statement (e.g., `display "@RBTOT is ", @RBTOT`).

Extended Logging may also be enabled with the `/x` command line switch.

@XLTFILE read/write
filename

Stores the name of the Translate File used in Terminal mode to filter, translate, or substitute characters (see “Translate File Format” on page 284).

Setup field: Translate File

@XONXOFF read/write
YES [NO]

Specifies whether software flow control is enabled. Not all computers support XON/XOFF flow control.

Setup field: XON/XOFF Pacing

@ZMAUTODOWN read/write
YES [NO]

For Zmodem transfers, specifies Auto Receive mode, in which downloading begins immediately after Filetransfer mode is entered.

Zmodem Protocol subwindow: Auto Receive

@ZMBLKLN read/write
[0] 24 – 1024

For Zmodem transfers, overrides the default block length, which is determined by the baud rate of the connection. The default, 0, specifies no limit to block length.

Zmodem Protocol subwindow: Limit Block Length

@ZMCONVR read/write
[ASCII] BINARY

For Zmodem transfers, specifies whether received files will be treated as ASCII or BINARY. For correct file conversion to ASCII, the remote computer must send the files as ASCII.

Zmodem Protocol subwindow: File Conversion

@ZMCONVSread/write
[NONE] ASCII BINARY

For Zmodem transfers, specifies whether files sent are to be treated as BINARY or ASCII, overriding the File Conversion setting of the receiving system. NONE specifies no override.

Zmodem Protocol subwindow: Conversion Override

@ZMCRCread/write
16 BITS **[32 BITS]**

For Zmodem transfers, specifies which CRC error detection is to be used.

Zmodem Protocol subwindow: CRC

@ZMCTLESCRread/write
YES **[NO]**

For Zmodem transfers, specifies whether all control characters received will be link-escape encoded for transparency.

Zmodem Protocol subwindow: Esc All Control Chars

@ZMCTLESCSread/write
YES **[NO]**

For Zmodem transfers, specifies whether all control characters sent will be link-escape encoded for transparency.

Zmodem Protocol subwindow: Esc All Control Chars

@ZMEXISTread/write
YES **[NO]**

For Zmodem transfers, specifies whether transfers will occur only if the file already exists on the destination system.

Zmodem Protocol subwindow: File Must Already Exist

@ZMFRMLENread/write
[0] 24 – 1024

For Zmodem transfers, specifies a limit for frame length that forces the sender to wait for a response from the receiver before sending the next frame. The default, 0, specifies no limit to frame length.

Zmodem Protocol subwindow: Limit Frame Length

@ZMMANAGRread/write
NONE PROTECT
[CLOBBER] APPEND

For Zmodem transfers, specifies a file management option for files received. See the File Management setup field (page 99) for a description of each option.

Zmodem Protocol subwindow: File Management

@ZMMANAGSread/write
[NONE] PROTECT
CLOBBER NEWER
NEWER/LONGER
DIFFERENT APPEND

For Zmodem transfers, specifies a file management option for files sent. See the Management Option setup field (page 96) for a description of each option.

Zmodem Protocol subwindow: Management Option

@ZMRESUMEread/write
YES **[NO]**

For Zmodem transfers, specifies continuation of an aborted file transfer from point of interruption. The destination file must already exist and must be smaller than the source file.

Zmodem Protocol subwindow: Resume Interrupted File

@ZMWINDOW

read/write
[0] – 9999

For Zmodem transfers, specifies the size of the transmit window.
The default, 0, specifies no limit to the size of the transmit window.

Zmodem Protocol subwindow: Size of Tx Window

Chapter 17

Data Stream Control and Terminal Emulation

Introduction

All versions of BLAST support data filtering and translation of incoming and outgoing data streams. This chapter describes these features as well as terminal emulation and keyboard mapping. Through terminal emulation, BLAST provides terminal functionality for a range of popular character terminals. With keyboard mapping, you can reassign the functions of the standard keyboard keys as well as the “BLAST keys” that control BLAST functions.

Data Stream Filtering and Alteration

BLAST allows for the translation, substitution, or filtering (removal) of individual characters in the data stream during terminal ses-

sions. This character manipulation can be used to do one of the following:

- ◇ Prevent the display of unwanted characters.
- ◇ Display international character sets.
- ◇ Prevent the transmission of certain key codes.
- ◇ Remap keys to send characters other than their defaults.
- ◇ Prevent characters from being saved in the capture file.
- ◇ Prevent characters from being sent with a file upload.

For example, Dow Jones News Service sends special start- and end-of-record characters that print non-ASCII characters on the screen. The standard translate file supplied with BLAST filters out these characters so that they do not appear on your display. If you wanted to automate your access to Dow Jones by writing a script, you might need to TTRAP for these filtered characters. For the TTRAP to see these characters, you would have to change the filter in order to allow these characters to pass.

Translate File Format

A copy of the standard translate file is on your distribution media as TRANSLAT.TBL. This file is distributed with the defaults used when the Translate File setup field (page 71) is empty. The translate file contains two tables: the receive table, which operates on characters received from the remote system, and the transmit table, which operates on characters sent to the remote system.

The receive and transmit tables within a BLAST translate file contain an array of 256 hexadecimal values. These values correspond to the 8-bit ASCII character set. The decimal value of a character ranging from 0 to 255 is used as an index to the character positions in the table. The hexadecimal value at that location in the table is substituted for the hexadecimal value of the original character.

TRANSLAT.TBL contains the following receive and transmit default tables:

:RECVTABL

-00,	-01,	-02,	-03,	-04,	-05,	-06,	07,
08,	09,	0A,	0B,	0C,	0D,	0E,	0F,
-10,	-11,	-12,	-13,	-14,	-15,	-16,	-17,
-18,	-19,	-1A,	-1B,	-1C,	-1D,	-1E,	-1F,
20,	21,	22,	23,	24,	25,	26,	27,
28,	29,	2A,	2B,	2C,	2D,	2E,	2F,
30,	31,	32,	33,	34,	35,	36,	37,
38,	39,	3A,	3B,	3C,	3D,	3E,	3F,
40,	41,	42,	43,	44,	45,	46,	47,
48,	49,	4A,	4B,	4C,	4D,	4E,	4F,
50,	51,	52,	53,	54,	55,	56,	57,
58,	59,	5A,	5B,	5C,	5D,	5E,	5F,
60,	61,	62,	63,	64,	65,	66,	67,
68,	69,	6A,	6B,	6C,	6D,	6E,	6F,
70,	71,	72,	73,	74,	75,	76,	77,
78,	79,	7A,	7B,	7C,	7D,	7E,	7F,
-00,	-01,	-02,	-03,	-04,	-05,	-06,	07,
08,	09,	0A,	0B,	0C,	0D,	0E,	0F,
-10,	-11,	-12,	-13,	-14,	-15,	-16,	-17,
-18,	-19,	-1A,	1B,	-1C,	-1D,	-1E,	-1F,
20,	21,	22,	23,	24,	25,	26,	27,
28,	29,	2A,	2B,	2C,	2D,	2E,	2F,
30,	31,	32,	33,	34,	35,	36,	37,
38,	39,	3A,	3B,	3C,	3D,	3E,	3F,
40,	41,	42,	43,	44,	45,	46,	47,
48,	49,	4A,	4B,	4C,	4D,	4E,	4F,
50,	51,	52,	53,	54,	55,	56,	57,
58,	59,	5A,	5B,	5C,	5D,	5E,	5F,
60,	61,	62,	63,	64,	65,	66,	67,
68,	69,	6A,	6B,	6C,	6D,	6E,	6F,
70,	71,	72,	73,	74,	75,	76,	77,
78,	79,	7A,	7B,	7C,	7D,	7E,	7F,

:XMITTABL

00,	01,	02,	03,	04,	05,	06,	07,
08,	09,	0A,	0B,	0C,	0D,	0E,	0F,
10,	11,	12,	13,	14,	15,	16,	17,
18,	19,	1A,	1B,	1C,	1D,	1E,	1F,
20,	21,	22,	23,	24,	25,	26,	27,
28,	29,	2A,	2B,	2C,	2D,	2E,	2F,
30,	31,	32,	33,	34,	35,	36,	37,
38,	39,	3A,	3B,	3C,	3D,	3E,	3F,
40,	41,	42,	43,	44,	45,	46,	47,
48,	49,	4A,	4B,	4C,	4D,	4E,	4F,
50,	51,	52,	53,	54,	55,	56,	57,
58,	59,	5A,	5B,	5C,	5D,	5E,	5F,
60,	61,	62,	63,	64,	65,	66,	67,
68,	69,	6A,	6B,	6C,	6D,	6E,	6F,
70,	71,	72,	73,	74,	75,	76,	77,
78,	79,	7A,	7B,	7C,	7D,	7E,	7F,
80,	81,	82,	83,	84,	85,	86,	87,
88,	89,	8A,	8B,	8C,	8D,	8E,	8F,
90,	91,	92,	93,	94,	95,	96,	97,
98,	99,	9A,	9B,	9C,	9D,	9E,	9F,
A0,	A1,	A2,	A3,	A4,	A5,	A6,	A7,
A8,	A9,	AA,	AB,	AC,	AD,	AE,	AF,
B0,	B1,	B2,	B3,	B4,	B5,	B6,	B7,
B8,	B9,	BA,	BB,	BC,	BD,	BE,	BF,
C0,	C1,	C2,	C3,	C4,	C5,	C6,	C7,
C8,	C9,	CA,	CB,	CC,	CD,	CE,	CF,
D0,	D1,	D2,	D3,	D4,	D5,	D6,	D7,
D8,	D9,	DA,	DB,	DC,	DD,	DE,	DF,
E0,	E1,	E2,	E3,	E4,	E5,	E6,	E7,
E8,	E9,	EA,	EB,	EC,	ED,	EE,	EF,
F0,	F1,	F2,	F3,	F4,	F5,	F6,	F7,
F8,	F9,	FA,	FB,	FC,	FD,	FE,	FF,

TRANSLAT.TBL can either filter, translate, or substitute characters.

Filtering – The default values of the receive table cause it to filter the following characters:

NUL (00)	ACK (06)	NAK (15)	ESC (1B)
SOH (01)	DLE (10)	SYN (16)	FS (1C)
STX (02)	DC1 (11)	ETB (17)	GS (1D)
ETX (03)	DC2 (12)	CAN (18)	RS (1E)
EOT (04)	DC3 (13)	EM (19)	US (1F)
ENQ (05)	DC4 (14)	SUB (1A)	

Values to be filtered from the transmitting or receiving data stream are preceded by a minus sign. A minus sign indicates that the value following it is ignored.

Translation – The default receive table also translates all “high” ASCII characters (8-bit characters above 127 [decimal] or 7F [hexadecimal] in value) to “low” ASCII (7-bit) characters by stripping the 8th bit. You will notice in the :RECVTABL illustrated above that the 17th row of the table begins, as does the 1st row, with “-00” and that the lower half of the table duplicates the upper half.

Substitution – A new hexadecimal value can be substituted for any existing default value in either the receive or transmit table. For example, suppose that you want to replace all upper case “A”s from the received data stream with lower case “b”s. You would:

- ◇ Find the character “A” in the ASCII table in Appendix D. You will see that the decimal value of “A” is 65 whereas the hexadecimal value is 41.
- ◇ Now find the hexadecimal value located in the 65th position of the translate table. Begin counting at the upper left-hand corner of the table (“-00” or “00”), moving from left to right and counting down the rows. Start your count from zero, and count until you reach the 65th position. The value in the 65th position is 41, the hexadecimal value for “A”.
- ◇ Look in Appendix D again and determine the hexadecimal value for “b”. That value is 62.
- ◇ Replace the value 41 in the translate table with 62. From now on, all “A”s in the received data stream will be translated to “b”s.

NOTE: The default *transmit* table transmits all characters without filtering, translation, or substitution.

Creating and Editing a Translate File

When specifying new values for a translate file, be sure not to delete an entry in the table completely. This will cause all entries in the table to shift values. To modify the file:

- ◇ Make a copy of the TRANSLAT.TBL file.
- ◇ Modify the new file using BLASTEDT or ASCII text editor.
Save the file in text format only.
- ◇ Locate the desired character position in the table and either enter a new value or place a minus in front of the existing value in the table.
- ◇ Save the new table where BLAST can access it. BLAST will look in the current directory first and then in BLASTDIR.

Specifying a Translate File in Your Setup

To specify a translate file for use during a session, type its name in the Translate File setup field.

International and Graphic Character Sets

For international and graphic character sets using TTY emulation, specify PASSTHRU.TBL in your Translate File setup field. This file takes into consideration those users requiring more than 128 characters to pass through without translation. Graphic terminals requiring special graphics characters using TTY emulation must also use this table. International and graphic character sets require 8-bit transparency to be set in the emulator. For DG emulation, set the Data Bits setup field of the DG Emulation subwindow to 8.

Terminal Emulation

A “terminal” is a video monitor and keyboard that has been custom configured to generate and respond to formatting codes used by a particular computer system. The VT100 terminal, for example, was originally designed to operate with a Digital Equipment Corporation VAX computer. Particular sequences of ASCII characters were defined to signal special actions, such as cursor movement, printer ac-

tivation, and screen display behavior. In order to use your PC as a terminal to a multi-user host like a VAX, it must be able to produce and respond to the host's terminal control codes—a process called “terminal emulation.” BLAST's VT100 terminal emulation allows your PC to operate like a VT100 terminal.

The following emulators are available in BLAST:

<u>BLAST Emulator</u>	<u>Terminal Emulated</u>
ADM3A	Lear Siegler ADM3A
D80	Ampex Dialog 80
D200	Data General D200
D411	Data General D411
D410	Data General D410
D461	Data General D461
HP2392	Hewlett-Packard 2392A
IBM3101	IBM 3101
ANSI	PC ANSI Color
TTY	Standard ANSI terminal
TV920	Televideo 920 series
VT52	DEC VT52
VT100	DEC VT100
VT220	DEC VT220
VT320	DEC VT320
WYSE50	Wyse 50
WYSE60	Wyse 60

Most of these terminals feature unique keys to perform certain functions, for example, the DO key on a VT220 terminal. Often, it is possible to assign a PC key to perform the same task as a special terminal key. In other cases, it may be necessary to assign a combination of PC keys to perform the function—the DO key is equivalent to ALT F5. Thus, your keystrokes are “mapped” or “routed” through BLAST's software to generate the required sequence of ASCII code for each terminal function. The default keyboard maps for all of BLAST's emulators are in Appendix B. With the remapping software (see “Keyboard Mapping Utility” on page 296 later in this chapter), you can reassign terminal function keystrokes.

The default emulator for a session is specified in the Emulation setup field. All of the emulators available except TTY and IBM3101 have subwindows that appear automatically when you press ENTER. See Chapter 5 for more information on these setup subwindows fields.

NOTE: To use 132-column compressed mode in VT and WYSE series emulations, your PC must be equipped with an EGA or VGA adapter card with full 132-column hardware support. Your monitor

must also support 132-column mode. Check the documentation for your video card and monitor to see if 132-column mode is supported. A description of the terminal emulations supported by BLAST follows:

TTY Emulation

TTY Emulation Using Default Translate Table

If you choose TTY emulation and leave the Translate File setup field blank, BLAST will use an internal default translate file similar to TRANSLAT.TBL. As noted in the “Data Stream Filtering and Alteration” section of this chapter, TRANSLAT.TBL filters the following characters:

NUL (00)	ACK (06)	NAK (15)	ESC (1B)
SOH (01)	DLE (10)	SYN (16)	FS (1C)
STX (02)	DC1 (11)	ETB (17)	GS (1D)
ETX (03)	DC2 (12)	CAN (18)	RS (1E)
EOT (04)	DC3 (13)	EM (19)	US (1F)
ENQ (05)	DC4 (14)	SUB (1A)	

The receive table also converts all 8-bit characters (above 7F in value) to 7-bit characters.

Transparent TTY Emulation Using Pass-Through Table

In typical terminal operation, the video display and the operating system pass four different types of information to each other:

- ◇ Readable text.
- ◇ Screen formatting commands to the display.
- ◇ Flow control between the display and the system.
- ◇ Interrupt and function codes.

When using BLAST to act as a terminal to a remote computer, all the different types of information must be passed from the display through the PC to the remote computer. No characters should be changed, discarded, or processed by the local computer. It must act as a “cable” between the PC display and the remote computer.

For complete 8-bit transparency, BLAST requires that you do the following:

- ◇ Enter the name `PASSTHRU.TBL` in the Translate File setup field. `PASSTHRU.TBL` is a file supplied with BLAST.
- ◇ Set the XON/XOFF Setup field to `NO`. (Flow control characters will be passed through, and flow control will be disabled.)
- ◇ Set the RTS/CTS Setup field for hardware flow control set to `YES` if supported by your system.
- ◇ Set the Emulator setup field to `TTY`.
- ◇ Set the Baud Rate setup field on both the local and remote computers to a setting lower than the baud rate between the local computer and the local terminal. This will insure that the remote computer cannot send lines of text faster than they can be displayed locally.

DEC VT320, VT220, VT100, and VT52 Emulation

The BLAST VT emulators provide precise emulation of the DEC VT320, VT220, VT100, and VT52 terminals. For a description of the setup options for these emulators, see “DEC VT Emulation Sub-windows” on page 74.

Supported Features

These emulators support the following features:

- ◇ All cursor positioning sequences and tab settings.
- ◇ All of the software-selectable operating states (or modes) available for the VT series of terminals, including standard ANSI and DEC private modes.
- ◇ The `USASCII`, `UK`, `FRENCH`, `GERMAN`, `ITALIAN`, `SPANISH`, and `DANISH` character sets. The default value is `USASCII` which, allows 8-bit data; the other character sets allow only 7-bit data.
- ◇ The DEC Supplemental Graphics and ISO LATIN-1 character sets.
- ◇ Scrolling regions, line and character editing, and character attribute commands.
- ◇ All print operations including Autoprint, Print Screen, and Printer controller (printer pass-through).

- ◇ 132-column compressed video modes on EGA/VGA systems.
- ◇ Horizontal scrolling control to accommodate 132-column display on a standard PC 80-column screen. The Scroll Left, Scroll Right, and Scroll mode keys may be used within Terminal mode and may be redefined with the BLASTKBD utility (see “Keyboard Mapping Utility” on page 296 later in this chapter). To set the default mode for the number of columns to scroll, specify the column width in the VT Emulation setup subwindow (page 74).

Special Considerations

The following features are not supported by these emulators:

- ◇ Smooth scrolling.
- ◇ Downloadable character sets. These will be ignored by the application.

The following features are supported in the specified limited manner:

- ◇ Double-width characters are handled by displaying a single-width character and a space in a double-width position.
- ◇ Double-height characters are displayed in the top half of a double-height position.

For the Key Definition Charts for these emulators, see “DEC VT320 and VT220 Keys” on page 330 and “DEC VT100 and VT52 Keys” on page 331.

ANSI Emulation

The BLAST ANSI emulator provides functional emulation of the IBM PC ANSI standard, including full color and extended attribute support. Choose ANSI for dialing Bulletin Board Systems or other computers that offer ANSI support. For a description of the BLAST ANSI emulation setup options, see “ANSI Emulation Subwindow” on page 72; for the Key Definition Chart for the BLAST ANSI emulator, see “ANSI Keys” on page 331.

DG D200, D410, D411, and D461 Emulation

The BLAST DG emulators provide precise emulation of the Data General D200, D410, D411, and D461 terminals. For a description of the setup options for these emulators, see “DG Emulation Subwindow” on page 73.

Special Considerations

There are a few restrictions to using Data General emulation in BLAST:

- ◇ ANSI mode is not supported—BLAST provides ANSI and VT320/220/100/52 emulators for ANSI emulation.
- ◇ User-defined character sets are not supported, and the sequences associated with defining and loading user-defined character sets are ignored.
- ◇ The maximum number of characters that define a pattern for a line is limited to 16 characters as opposed to 32 characters in the set pattern command.
- ◇ The scrolling rate commands are ignored.
- ◇ 135 column displays are not supported.

For the Key Definition Chart for these emulators, see “Data General D461, D411, D410, D200 Keys” on page 332.

WYSE50/60, TV920, D80, and ADM3A Emulation

The BLAST WYSE50/TV920/D80/ADM3A and WYSE60 emulators are functionally equivalent to the WYSE 50+, Televideo family, Ampex D80, Lear Siegler ADM 3A, and WYSE 60 terminals, respectively. For a description of the setup options for these emulators, see “WYSE Emulation Subwindows” on page 78.

Supported Features

These emulators support the following terminal features:

- ◇ 24 or 25 data lines.
- ◇ User selectable status line.
- ◇ 80- or 132- column modes. (132-column mode is represented in a compressed fashion if hardware support is available).
- ◇ Multiple pages (up to 4; the default is 1).
- ◇ Split screens.
- ◇ Normal, dim, blink, blank, underline, and reverse attributes.

- ◇ Protected fields.
- ◇ Graphics characters.
- ◇ Print functions.
- ◇ Editing functions.
- ◇ Ability to choose PC Term or WYSE Personality (WYSE60 only).
- ◇ 8-bit data support (WYSE60 only).

For the Key Definition Chart for these emulators, see “WYSE60, WYSE50, TV920, D80, and ADM3A Keys” on page 333.

HP2392 Emulation

The BLAST HP2392 emulator is functionally equivalent to the Hewlett Packard 2392 terminal. For a description of the setup options for the BLAST HP2392 emulator, see “HP Emulation Subwindow” on page 82

Supported Features

The following features of the HP 2392A series are supported within BLAST:

- ◇ Terminal identification for terminal ID requests.
- ◇ Handshaking options.
- ◇ Automatic text wrapping.
- ◇ BACKSPACE erase.
- ◇ Block mode operation.

For the Key Definition Chart for the BLAST HP2392 emulator, see “HP2392 Keys” on page 334.

IBM3101 Emulation

The BLAST IBM3101 emulator is functionally equivalent to the IBM 3101 series terminal.

Supported Features

The following features of the IBM 3101 series are supported within BLAST:

- ◇ Character mode operation.
- ◇ Block mode operation.
- ◇ Programming mode operation.
- ◇ Most dip switch functions (exceptions are noted below).

These features of the IBM 3101 series are not supported within BLAST:

- ◇ Transparent mode operation.
- ◇ Local mode operation.
- ◇ International support ‘dead keys.’
- ◇ The following dip switch functions: Send Line option, Number of Time Fill characters, Reverse Video on/off, Blink Cursor on/off, Line Speed for Auxiliary Interface.

For the Key Definition Chart for the BLAST IBM 3101 emulator, see “IBM3101 Keys” on page 335

NOTE: BLAST uses a default configuration for the 3101 dip switches unless the user specifies overriding values in a file called “CONFIG.31.” If CONFIG.31 does not exist when the user goes into Terminal mode, the dip switch settings will default to the following values:

MODE: CHAR	(character vs. block mode)
LTA: CR	(line turnaround character is a carriage return)
NULL_SUPP: ON	(null suppression is on)
AUTO_NL: ON	(auto newline/autowrap is on)
AUTO LF: OFF	(auto linefeed is off)
CR_LF: OFF	(carriage returns are not followed by line feeds)
SCROLL: OFF	(end-of-screen scrolling is off)

To customize the dip switch configuration, create the CONFIG.31 file using the format <name : value. The possible names and values are:

MODE	ON or OFF
LTA	CP, EOT, ETX, or XOFF
NULL_SUPP	ON or OFF
AUTO_LF	ON or OFF
CR_LF	ON or OFF (ON implies a line feed with a CR)
SCROLL	ON or OFF

For example, the following command substitutes an XOFF as the line turnaround character and forces scrolling at the end of the screen.

```
<LTA: XOFF
```

```
<SCROLL: ON
```

The 3101 half-duplex dip switch may be configured by toggling the Local Echo setup field to ON for half duplex and OFF for full duplex.

Transparent Print/Auxiliary Print

BLAST supports Transparent Print mode (data redirected to an attached printer as well as displayed on the screen) under the VT, ANSI, and WYSE60/50 emulations. Additionally, WYSE 60/50 emulations support Auxiliary Print mode (data redirected to an attached printer only).

BLAST recognizes the following codes for these functions:

WYSE60/50

Transparent Print mode on:	ESC d #
Auxiliary Print mode on:	CTRL R
Transparent and Auxiliary Print mode off:	CTRL T

ANSI/VT

Transparent Print mode on:	ESC [5i
Transparent Print mode off:	ESC [4i

BLAST defaults to LPT1: for these print functions.

Keyboard Mapping Utility

Computer users sometimes encounter difficulties when emulating a terminal. For example:

- ◇ A key sequence meant to be passed to the remote computer is instead intercepted by an application (or the operating system).
- ◇ An emulator keymap is awkward for a particular application.
- ◇ Repetitive keystrokes are required for a particular application.
- ◇ A required key does not exist on the user's keyboard.

The keyboard mapping utility **BLASTKBD** helps address these problems. In **BLASTKBD**, there are three types of specially assigned key subsets in the **BLAST** key set: **Soft Keys**, **BLAST Keys**, and **Hot Keys**. In addition, **BLASTKBD** includes **Emulator Maps** and **User-Defined Maps**. Below is a brief description of each, followed by sections giving instructions on mapping and/or remapping each key set:

Soft Keys – allow you to send often-used character strings to a remote system with a single keystroke. The use of **Soft Keys** is described later in this chapter.

BLAST Keys – allow you to use special key sequences to differentiate between local commands and characters meant for the remote system. The **BLAST Keys** perform local functions, such as exiting Terminal mode. The **BLAST Keys** are listed in Appendix B.

Hot Keys – allow access to often-used functions from Terminal, Filetransfer, and Access modes. **Hot Keys** are essentially macros that activate **BLAST** menu commands and return you to your starting point with just a few keystrokes. Typing **ALT F** from a console in Terminal mode, for example, starts Filetransfer mode and automatically returns you to Terminal mode when file transfer is completed. The **Hot Keys** are listed in Appendix B.

Emulator Maps – keyboard maps for emulators within the **BLAST** program. With **BLASTKBD**, you can reroute existing functions to different keys on your keyboard. For a list of keys for the existing emulators, see “Terminal Emulation Keys” on page 329.

User-Defined Maps – keyboard maps that can be created for different applications, keyboards, or users. Unlike the emulator maps, user-defined maps can specify functions as well as keys.

Running BLASTKBD

You can start BLASTKBD by typing “blastkbd” on the command line or, during a terminal session, by pressing *ATTN M* or *ATTN E*. *ATTN M* will display the main selection window (Figure 17-1) whereas *ATTN E* will take you directly to the specific map subwindow for the current emulator. From the emulator map, pressing *ESC* will return you to Terminal mode if that is where you started. If you started BLASTKBD from the command line, pressing *ESC* from the main BLASTKBD window will return you to the command line.

After you have edited a keyboard map or one of the BLAST special key sets, press *S* to save your changes. Press *ESC* if you wish to exit without saving the changes.

To select a BLAST special assigned key subset or a map from the BLASTKBD main window, use the commands described at the bottom of the window to highlight the desired selection and press *ENTER*.

FIGURE 17-1

```
Keyboard Mapping Utility

BLAST keys:    SOFTKEYS
               BLAST
               HOTKEYS
Emulator maps: VT320/VT220
               VT100/VT52
               ANSI
               D461/D411/D410/D200
               WYSE60/WYSE50/TV920/D80/ADM3A
               IBM3101
               HP2392
User defined maps: 3708AT
                  7171AT
                  HYDRAII
                  HYDRASNA
                  RENEXAT
                  3708101
                  RENEX101
                  7171101
                  PCANSI

UP: ↑          ADD: 'a'      SELECT: <enter>
DOWN: ↓       DEL: 'x'      QUIT: <ESC>
```

NOTE: In the subwindows discussed below, some characters cannot be entered merely by pressing the corresponding key on the keyboard. The following table indicates how these characters are to be entered:

ESC	CTRL [
TAB	CTRL I
ENTER	CTRL M

For example, to include the escape character in a key sequence, press *CTRL [* instead of pressing *ESC*. Some characters may appear in octal form, for example, *CTRL^* may appear as *\036*.

Soft Keys

Many terminals offer a way of storing a set of often-used character strings that can be sent to the remote system with a single keystroke. BLAST provides this capability with Soft Keys. If you highlight Soft Keys in the main window and press `ENTER`, the Soft Key window (Figure 17-2) will appear.

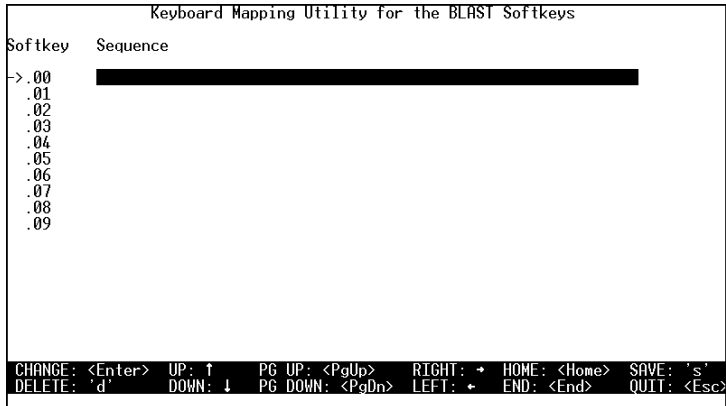


FIGURE 17-2

To create a Soft Key, highlight the sequence for the Soft Key you have selected (0–9) and enter the text string to be sent to the remote system when that Soft Key is pressed. Each string can be up to 69 characters long.

BLAST allows ten Soft Keys. A Soft Key is activated from within Terminal mode with the following combination:

```
ATTN Soft_Key_number
```

where *Soft_Key_number* is the number key corresponding to the number of the text string. For example, 0 corresponds to .00 text string, 1 to the .01 text string, and 2 to the .02 text string.

BLAST Keys

You can also use `BLASTKBD` to modify the BLAST key subset. When you select BLAST from the BLAST Key set in the `BLASTKBD` main window and press `ENTER`, the BLAST Key sub-window (Figure 17-3, next page) will appear.

FIGURE 17-3

Keyboard Mapping Utility for the BLAST Keys			
Function	Key 1	Key 2	Key 3
->Cursor Up	<u>↑</u>	<ext><up>	
Cursor Down	<down>	<ext><down>	
Cursor Left	<left>	<ext><left>	
Cursor Right	<right>	<ext><right>	
Home	<home>	<ext><home>	
End	<end>	<ext><end>	
Page Up	<pgup>	<ext><pgup>	
Page Down	<pgdn>	<ext><pgdn>	
ESC key	<esc>		
Del Char		<ext>	
Insert Mode	<ins>	<ext><ins>	
Help	<f1>		

CHANGE: <Enter>	UP: ↑	PG UP: <PgUp>	RIGHT: →	HOME: <Home>	SAVE: 's'
DELETE: 'd'	DOWN: ↓	PG DOWN: <PgDn>	LEFT: ←	END: <End>	QUIT: <Esc>

There are four columns—the first displays the functions supported by the BLAST Keys and the other three contain the key sequences you choose to perform that function. Up to three key sequences may be specified for the same function. To remap a function, highlight one of the three key sequences to the right of the function and press ENTER. The message “Press any key to remap function...” is displayed. Type the key (or combination of keys) that will serve as this function. Repeat this process until you have remapped all the functions that you want; then press S to save your remappings and return to the BLASTKBD main window.

NOTE: BLAST keys do not change from emulator to emulator. For example, if you map the Cursor Down function as CTRL 2 in the BLAST Keys subwindow while using the VT320 emulator, that sequence will also perform the same function if you switch to the WYSE60 emulator.

You cannot use BLASTKBD to remap Attention (ATTN) Key sequences. The Attention Key can be remapped via the Attention Key setup field (page 72).

Hot Keys

Hot Keys allow access to often-used functions from Terminal, Filetransfer, and Access modes. If you select Hot Keys from the BLAST Key set in the BLASTKBD main window and press ENTER, the Hot Key subwindow will appear (Figure 17-4, next page). Hot Keys override *all* other functions. For example, if you map both the VT Find key and the Filetransfer Hot Key to ALT F, pressing ALT F will *always* start Filetransfer mode and *never* act as the VT Find key.

FIGURE 17-4

Keyboard Mapping Utility for the HOTKEYS Keys			
Function	Key 1	Key 2	Active Mode
>Background	<alt>n		Always (if /b)
Abort BLAST	<alt>x		Terminal
Connect	<alt>c		Terminal
Disconnect	<alt>d		Terminal
Capture	<alt>o		Terminal
Learn	<alt>r		Terminal
Select Setup	<alt>s		Terminal
Modify Setup	<alt>m		Terminal
New Setup	<alt>n		Terminal
Write Setup	<alt>w		Terminal
Access	<alt>a		Terminal
Local Edit	<alt>e		Terminal, FT
Local Print			Terminal, FT
Local Type			Terminal, FT
Local List	<alt>l		Terminal, FT
Local View			Terminal, FT, Access
Local System			Terminal, FT, Access
CHANGE: <Enter> UP: ↑ PG UP: <PgUp> RIGHT: → HOME: <Home> SAVE: 's'			
DELETE: 'd' DOWN: ↓ PG DOWN: <PgDn> LEFT: ← END: <End> QUIT: <Esc>			

To map or remap a function, highlight the first key sequence to the right of the function and press enter. The message “Press any key to remap function...” is displayed. Type the key (or combination of keys) that will serve as this function. Repeat this process until you have remapped all the functions that you want; then press **s** to save your remappings and return to the BLASTKBD main window.

NOTE: A Hot Key can only be mapped to a single keystroke. Any keystrokes entered into the second column will be ignored by BLAST.

Emulator Maps

Emulator maps act as links between your keyboard and the terminal you are emulating. For example, if you are using an AT extended keyboard through the VT320 emulator to a VAX minicomputer, the keymap will link the **F1** key to the VT320 **PF1** function. To select an emulator, highlight the emulator in the BLASTKBD main window and press **ENTER**; the emulator subwindow will then appear. For example, if you select VT320/VT220, the subwindow in Figure 17-5 (next page) will appear.

To remap a function, highlight one of the three key sequences to the right of the function and press **ENTER**. The message “Press any key to remap function...” is displayed. Type the key (or combination of keys) that will serve as this function. Repeat this process until you have remapped all the functions that you want; then press **s** to save your remappings and return to the BLASTKBD main window. Up to three key sequences maybe specified for the same function.

FIGURE 17-5

Keyboard Mapping Utility for the VT320/VT220 Keys			
Function	Key 1	Key 2	Key 3
->Backspace	<ctr><back>		
Del	<back>		
Cursor Up	<up>	<ext><up>	
Cursor Down	<down>	<ext><down>	
Cursor Left	<left>	<ext><left>	
Cursor Right	<right>	<ext><right>	
Keypad 0	<keypad>0		
Keypad 1	<keypad>1		
Keypad 2	<keypad>2		
Keypad 3	<keypad>3		
Keypad 4	<keypad>4		
Keypad 5	<keypad>5		
Keypad 6	<keypad>6		
Keypad 7	<keypad>7		
Keypad 8	<keypad>8		
Keypad 9	<keypad>9		
Keypad -	<keypad>-		
CHANGE: <Enter> UP: ↑ PG UP: <PgUp> RIGHT: → HOME: <Home> SAVE: 's'			
DELETE: 'd' DOWN: ↓ PG DOWN: <PgDn> LEFT: ← END: <End> QUIT: <Esc>			

User-Defined Maps

A powerful feature of BLASTKBD is the option to create your own keyboard maps for different applications, keyboards, or users. For example, you can customize a map for a remote database application and save it under the name “data,” ready for use with BLAST. Once you have finished working with the database, you can load another map for another application.

To create a map, press A at the BLASTKBD main window. You will be prompted for the name of the new map. Pressing ENTER after typing in the name of the new map will add the map name to the list of maps in the BLASTKBD main window (the map name will also appear as a selection in the Keyboard File setup field). Pressing ENTER again will display the mapping subwindow (Figure 17-6). Unlike the emulator maps, user-defined maps allow you to specify the function as well as the keys.

FIGURE 17-6

Keyboard Mapping Utility for the 3708AT Keys			
Function	Key 1	Key 2	Key 3
->Backspace	<back>		
BackTab	<shift><tab>		
Clear	<ctrl>C		
Cursor Up	<up>		
Cursor Down	<down>		
Cursor Left	<left>		
Cursor Right	<right>		
Delete			
Enter	<enter>		
Erase EOF	<ctrl>E		
Insert	<ins>		
PA1	<alt>1		
PA2	<alt>2		
PA3	<alt>3		
PF1	<f1>		
Sequence: ^[D			
ADD FUNC: 'a' CHANGE: <Enter> UP: ↑ RIGHT: → HOME: <Home> SAVE: 's'			
DEL FUNC: 'x' DEL KEY: 'd' DOWN: ↓ LEFT: ← END: <End> QUIT: <Esc>			

The first step in assigning a function is to type the name of the function. If no functions have been assigned, simply type the name of the function in the field highlighted. To add a function, type A and then the name of the function you would like to add.

After typing the name of the function, press ENTER. The first key sequence will automatically be highlighted. Type the key sequence for the function you have just added and press ENTER. At the bottom of your screen, you will be prompted for the ASCII control sequence. Type either the ASCII control sequence or octal value for that function (for a list, see Appendix D) and press ENTER.

If you would like to add a second key sequence for the function or change an existing key sequence, highlight the desired key sequence to the right of the function and follow the same steps as you followed in entering the first key sequence. After you have finished mapping functions, press S to save your map.

Keyboard Map Selection in the Setup

All maps that you create are saved in a file called BLAST.TDF. Each time that you start BLAST, it will search the current directory for BLAST.TDF. If it cannot be located, BLAST then checks BLASTDIR. You can easily assign separate keymaps for several users or applications by copying different BLAST.TDF files into each directory. When you run BLAST from within an application directory, the proper BLAST.TDF file will automatically be loaded.

To select a specific user map from within a given .TDF file, highlight the Keyboard File setup field and use the SPACEBAR to cycle through the map choices. If you would like a map to be loaded automatically on startup, save it as a part of the setup.

Chapter 18

Remote Control

What Is Remote Control?

The Remote Control features of BLAST allow you to access and control a remote PC's screen, keyboard, disk drives, and printer. Remote control is ideal for troubleshooting remote sites, training and supporting PC operators, using your office databases or spreadsheets from home—any time you need complete control over a remote PC.

This chapter introduces basic concepts and guides you through the process of configuring BLAST for remote control. The *BHOST User Manual* included with the BLAST package describes how to set up the remote computer for control by BLAST.

Remote control allows one PC (the Control PC) to completely control another PC (the Host PC). The two PCs may be attached to modems communicating over a telephone circuit, hardwired together with a null modem cable, or connected to a Local Area Network.

The Control PC can run programs on the Host PC's hard drive, print documents, edit files, and more, as if the user were typing on the Host PC's keyboard. All video output and graphics (Hercules, CGA,

MCGA, EGA, and VGA) are displayed simultaneously on both PCs, with automatic translation between different video modes.

The Host PC

The Host PC runs a special program called BHOST, which operates in the background and is transparent to the Host PC user. BHOST “watches” the communications port and, when a call comes in, prompts the caller for a user identification and password. Once the caller is logged in, BHOST makes the Host PC’s services available to the Control PC.

The Host PC has access to a number of security features, including login accounts, multiple control levels, call-back security, and a log file to record system activity.

BHOST also offers background file transfers. The Control PC may transfer files to and from the Host PC while the Host PC user continues to work on other applications.

Nearly all of the configuration for a remote control session takes place on the Host PC through SETBHOST, a special administration that sets system defaults and keeps track of login accounts.

The Control PC

For the Control PC, Dialing into a Host PC is just like an ordinary terminal session except that, once connected, the Control PC selects Access from the Online menu. Access mode allows complete control of the Host PC and high-speed file transfers using the BLAST session protocol. (File Transfer Only and Terminal modes, discussed later, offer more limited control.)

In Access mode, the Control PC has access to a number of security features, including the ability to disable the Host PC’s keyboard, mouse, and screen during a session to prevent unauthorized operation.

Connecting to the Host PC

Connecting to the Host PC is the same as connecting to any other remote system. BLAST can automatically dial the phone (or connect to the network) and send your login ID and password to the Host PC. You may also perform this process manually.

Be sure that BHOST has been installed and configured on the Host PC before attempting to connect. See the *BHOST User Manual* for more information on installing and configuring BHOST.

Creating a BLAST Setup for BHOST

To automate your connection to a Host PC, create and save a new BLAST setup for your sessions with the Host PC (see Chapter 5 for a detailed description of setups). In the new setup:

- ◇ If you are using a modem, set the Phone Number to the phone number of the Host PC.
- ◇ Set the System Type field to BHOST if your BHOST account requires a login ID and password; set System Type to PC or NONE if your BHOST account does not require a login ID or password.
- ◇ If your BHOST account requires a login ID and password, enter these into the Userid and Password setup fields, respectively, exactly as they appear in SETBHOST on the Host PC. *These fields are case-sensitive.*
- ◇ Set Emulation to TTY or VT320.
- ◇ Set Protocol to BLAST.
- ◇ Set Packet Size to at least 200, BHOST's minimum setting; the maximum setting is 4085.
- ◇ In the BLAST protocol setup subwindow, set Compression Level according to the type of data you will transfer. Note that BHOST's compression level defaults to 1. Any additional compression is determined by the amount of memory allocated by a COMBUF assignment in BLAST.OPT on the Host PC. BHOST supports compression levels 0–4.
- ◇ If you are making a node-to-node connection across a LAN, set the Connection field to the same value as that in the Connection field of the BLAST Host setup on the Host PC.

Making the Connection and Logging On

Highlight the new Host PC setup in your Dialing Directory and select Connect from the Offline menu. BLAST will dial the modem (or connect to the network), log you onto the Host PC, and return to the Online menu.

NOTE: If your BHOST Account is set to Dial Back, BLAST will not return to the Online menu immediately. Instead, BHOST will disconnect after you log in and then dial your phone number from the Host PC. Once the connection has been re-established, BLAST will return to the Online menu.

Taking Control

How you take control of the Host PC depends on the Control mode setting in your BHOST Account. The possible settings are *Access*, *File Transfer Only*, and *Terminal*. The default Control mode, *Access*, provides complete control over the Host PC.

Access Mode – If your Control mode is set to *Access*, then press **A** from the Online menu to enter Access mode. You will then have complete control over the Host PC. All of your keystrokes are sent to the Host PC, and all of the Host PC's screen displays are sent to your PC. Access mode offers a number of powerful features. See “Using Access Mode” on page 308 for complete details.

Terminal Mode – If your Control mode is set to *Terminal*, you have limited control through the BLAST Remote menu in addition to limited Terminal access. To enter Terminal mode, press **T** from the Online menu. You will be limited to ASCII text display. Programs using graphics or full-screen text modes will execute, but the screen display will be corrupted and no error detection will be performed. Terminal mode requires special keyboard sequences to send control characters. See “Using Terminal Mode” on page 311.

File Transfer Only Mode – If your Control mode is set to *File Transfer Only*, then press **F** from the Online menu to enter BLAST Filetransfer mode and then **R** to access the Remote menu.

Online Menu Options

All three Control modes give you access to the following Online menu options: *Connect*, *Capture*, *Filetransfer*, *Script*, and *Disconnect*. In addition, with *Access* Control mode, you may select the *Access* menu option (see “Using Access Mode” on page 308), and with *Terminal* Control mode, you may select *Terminal* menu option (see “Using Terminal Mode” on page 311).

NOTE: `TERMINAL`, `TSEND`, `TTRAP`, and `TUPLOAD` commands can only be used with Terminal Control mode.

Transferring Files to and from the Host PC

With all three control modes, BLAST protocol is available for transferring files to and from the Host PC. Your transfers will take place in the background on the Host PC, transparent to the Host PC user.

Starting Filetransfer Mode

There are several ways to initiate a file transfer to or from the Host PC. In each case, the BLAST Filetransfer menu appears, and you will be able to Send and Get files and execute operating system commands from the Local and Remote menus.

- ◇ **From the Online menu** — Press `F` to start Filetransfer mode. Use this method if your BHOST account is set to `File Transfer Only`.
- ◇ **From Access mode** — Press the `ALT F` Hot Key, or press `ATTN ESC` to return to the Online menu and then press `F` to start Filetransfer mode.
- ◇ **From Terminal mode** — Press `ESC CTRL X` to start Filetransfer mode on the Host PC; then use one of the above methods to start Filetransfer locally. Alternatively, you may script file transfers.

Transferring Files

You may transfer files interactively (see “Performing Filetransfer Commands” on page 110) or via a BLAST script (see “File Transfers with BLAST Session Protocol” on page 178).

Ending Filetransfer Mode

When you have finished transferring files, press `ESC` to end Filetransfer mode. If you started Filetransfer mode with a Hot Key, you will be returned to Access or Terminal mode. Otherwise, you will be returned to the Online menu.

Disconnecting from the Host PC

From Access mode – Press *ATTN* ESC to return to the Online menu.

From File Transfer Only mode – Press ESC to return to the Online menu.

From Terminal mode – Press ESC CTRL L to log off of the Host PC and then press *ATTN* *ATTN* to return to the Online menu. Select the Disconnect command to disconnect from the Host PC.

Using Access Mode

If your BHOST account is set to *Access* Control mode, you may enter Access mode from the Online menu by selecting *Access*. Your screen will display the remote terminal screen—all you have to do is type commands as if you were seated at the Host PC.

The Access Menu

From Access mode, you may display the Access menu (Figure 18-1) by pressing *ATTN*.

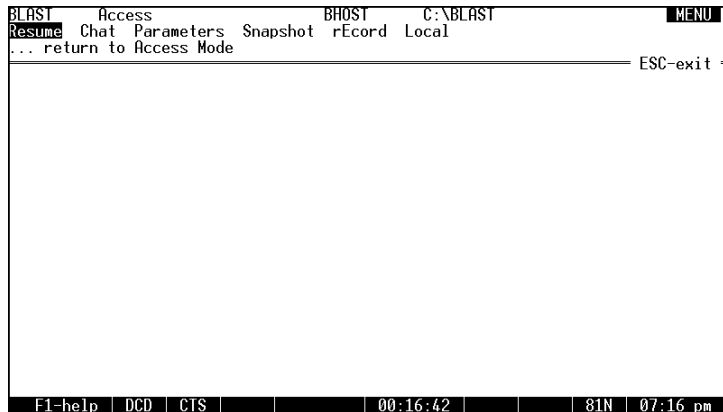


FIGURE 18-1

The Access menu provides the following features:

- ◇ Split-screen Chat mode, for communicating interactively with the Host PC user.

- ◇ Two camera modes, one for taking “snapshots” of individual screens and one for recording “movies” of your session.
- ◇ A simple menu for fine-tuning your remote control settings.
- ◇ Hot Keys to start file transfers, exit to a DOS shell, reboot the Host PC, and more.

To select a command from the Access menu, press the capitalized letter in the command name or move the cursor over the command and press ENTER. Following is a description of each command:

Resume – Press R to return to Access mode.

Chat – Press C to start Chat mode. Chat mode allows the Host and Controller to type messages to each other on the Chat screen (Figure 18-2), which is displayed on both the Controller and the Host screens. Either side may initiate a Chat unless the Host keyboard has been disabled. Once the Controller initiates a Chat, a disabled Host keyboard becomes active for the duration of the Chat.

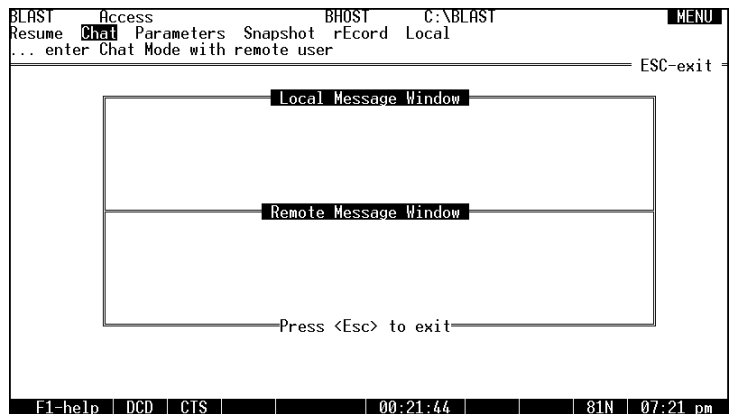


FIGURE 18-2

The Chat screen contains two windows, one for the Controller’s messages and one for the Host’s messages. Both sides may type at the same time. Chat mode will terminate when either user presses ESC.

Parameters – Press P to display the Session Parameters window containing parameter fields that can be adjusted to improve BHOST performance (see “Session Parameters Window” on page 313 for details).

Snapshot – Press **S** to take a snapshot of the current screen. You will be prompted for a filename. After typing the filename and pressing **ENTER**, you will be returned to the Access menu. The current screen image will be saved to your current directory. If you type in a filename without an extension, **BLAST** automatically uses the extension “.001.” Then, each time you take another snapshot, **BLAST** increments the extension by one (up to .099) and prompts you to save the new file.

BLAST saves text screens in standard ASCII file format and graphic screens in the .PCX format, which can be displayed with the **View** command from the **Local** menu or by a variety of third-party applications.

Record – Press **E** to record a “movie” of the screen appearance during your session (except **Chat** mode displays). You will be prompted for a filename. Type the filename and press **ENTER**. **BLAST** will then begin recording from the **Host PC** similar to **VCR** recording from a television. Escaping from the remote session screen for any reason will terminate the movie.

Movies can be replayed with the **View** command from the **Local** menu. By default, a movie is replayed at the same speed at which it was recorded. Press the up or down cursor keys during replay to speed up or slow down the movie. Note that movies can take up large amounts of disk space.

Local – Press **L** to display the **Local** menu for local **PC** commands. This command is identical to the **Local** command available from the **Offline** and **Online** menus (see “The **Local** Menu” on page 57 for details).

Access Mode Hot Keys

The following subset of the regular **BLAST** Hot Keys are active during **Access** mode:

<u>Function</u>	<u>Default Key Sequence</u>
Chat mode	*
Background	ALT H
Local View	*
Local System	*
Remote Reboot	*
Snapshot	*

<u>Function</u>	<u>Default Key Sequence</u>
Parameters	*
Record a movie	*
Filetransfer mode	ALT F

* To avoid potential conflicts with the programs running on the Host PC, these keys do not have default values. When you assign keys through the BLASTKBD utility, remember that the values you pick will not be available to the Host PC programs. For example, if you assign the ALT V key combination to the Local View function, then ALT V will never be sent to the Host PC, because it will be interpreted as a local command.

See “Hot Keys” on page 299 and Appendix B for more information on remapping Hot Keys.

Using Terminal Mode

Accessing and Logging Off Terminal Mode

If your BHOST account is set to Terminal Control mode, you may enter Terminal mode by choosing Terminal from the Online Menu. You will be prompted for a login ID and password on the terminal screen. You can return to the Online menu at any time by pressing *ATTN ATTN*.

In Terminal mode, a remote user can run programs with line-mode ASCII text displays. Programs using graphics or full-screen text modes will execute, but the screen display will be corrupted and no error detection will be performed.

When you are ready to log out, *you must log out of Terminal mode correctly*: Press ESC CTRL L—you will automatically be logged out of BHOST on the Host PC. You can then return to the Online menu by pressing *ATTN ATTN*; then hang up the modem by selecting Disconnect.

Terminal mode via BHOST is useful for accessing Host PCs from minicomputers or other computers that are not running BLAST. Any communications software that is capable of dialing the Host PC can connect with BHOST and act as a terminal to the Host PC.

Escape Sequences

Terminal mode requires special escape sequences to represent certain keys to the Host PC:

<u>PC Key</u>	<u>Escape Sequence</u>
Left Arrow	ESC F
Right Arrow	ESC G
Up Arrow	ESC T
Down Arrow	ESC V
Home	ESC H
End	ESC E
Page Up	ESC P
Page Down	ESC Q
Insert	ESC I
Delete	ESC D
Numeric Keypad 5	ESC .
Numeric Keypad *	ESC *
Break	ESC S
Caps Lock	ESC K
Num Lock	ESC N
Numeric Keypad +	ESC +
Numeric Keypad -	ESC -
F1–F10	ESC 1 – ESC 0
Esc	ESC ESC
All keys released	ESC SPACE
Ctrl	ESC C
Alt	ESC A
Left Shift	ESC Z
Right Shift	ESC /

The following escape sequences send special commands to BHOST:

<u>PC Key</u>	<u>Escape Sequence</u>
Filetransfer mode	ESC CTRL X
Repaint Screen	ESC CTRL R
Open Session Command window	ESC CTRL M
Log off	ESC CTRL L

Modifying BHOST Settings

There are three different ways to alter BHOST parameter settings—through SETBHOST, the Session Parameters Window, or the Session Command Window.

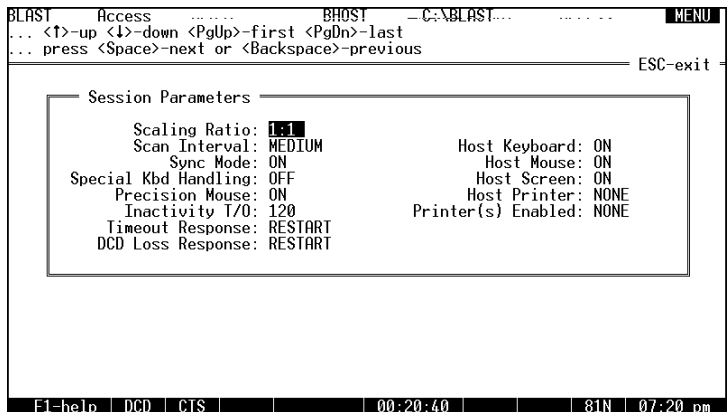
SETBHOST

You may alter BHOST parameter settings by starting SETBHOST on the Host PC by typing SETBHOST at the HOST PC's DOS prompt. For details on configuring BHOST via SETBHOST, see the *BHOST User Manual*. Note that the new settings will not take effect until BHOST has been restarted.

Session Parameters Window

If you are in Access mode and have a Superuser account, you may alter BHOST session parameters by choosing Parameter from the Access Menu. The Session Parameters window will then appear as shown in Figure 18-3 below. Move through the fields by pressing the arrow keys; move through the options of a field by pressing SPACE or BACKSPACE.

FIGURE 18-3



If you have a User account, you may change all of the settings on the Session Parameters screen except Inactivity T/O, Timeout Response, and DCD Loss Response. If you have a Restricted account, you cannot change any of the session parameter settings. If you change any of the settings via the Session Parameters window, the new settings will be in effect only for the duration of the session. Following is a description of the Session Parameter fields.

Scaling Ratio

[1:1] 1:4 1:16 1:64

Specifies how the Host PC's graphics are scaled for screen updates. BHOST usually sends the entire Host screen to the Control PC. The Scaling Ratio allows certain portions of the screen to be omitted, resulting in much faster performance. Scaling Ratio only applies to graphics screens.

When Scaling Ratio is set to a value other than 1 : 1, BHOST divides the Host PC screen into square grids and sends only the value of the first pixel in the grid. The Control PC then substitutes that value for each of the remaining pixels in the grid. For example, when Scaling Ratio is set to 1 : 4, BHOST sends only the first pixel of a 4-pixel grid. The Control PC writes that value for all four of the pixels in the grid.

1 : 1 – the entire Host screen is sent to the Control PC.

1 : 4 – the Host PC sends 1 pixel from a 4-pixel grid. (25% of the Host PC screen).

1 : 16 – the Host PC sends 1 pixel from a 16-pixel grid. (6.25% of the Host PC screen).

1 : 64 – the Host PC sends 1 pixel from a 64-pixel grid. (1.5% of the Host PC screen).

Use a higher Scaling Ratio (1 : 4, 1 : 16, or 1 : 64) when you want to see screens as quickly as possible and image quality is not important. For example, to move quickly through the opening screens of graphics applications, set Scaling Ratio to 1 : 64 as soon as you log onto the Host PC; then set it to 1 : 4 or 1 : 16 for fast updates, or to 1 : 1 for exact screens.

Scan Interval NONE HIGH [MEDIUM] LOW

Specifies how often BHOST scans the Host PC's display to see if the display has changed since the last scan. If it has, BHOST rescans the display and sends the new screen to the Control PC.

The higher the Scan Interval, the more often the display is updated. A higher Scan Interval, however, usually means slower program speed since the foreground application on the Host PC must be interrupted for the scan, and each image must be sent to the Control PC.

HIGH – The Host screen is scanned 18.2 times per second (after each PC clock tick).

MEDIUM – The Host screen is scanned twice per second (after each 8 PC clock ticks).

LOW – The Host screen is scanned once per second (after each 18 PC clock ticks).

NONE – The Host screen is scanned only when the operating system is not updating the screen.

Sync Mode **[ON] OFF**

Specifies whether the Host PC and the Control PC screens will be synchronized. When this field is set to ON, the Host PC screen is frozen while screen updates are sent to the Control PC. This mode completely synchronizes the two displays, but it slows the application speed. When this field is set to OFF, the Host PC screen is not frozen, resulting in significantly faster performance. The Control PC, however, may miss some intermittent screen images.

Special Kbd Handling **ON [OFF]**

Enables/disables Special Keyboard mode, which is necessary for machines running applications that intercept keyboard interrupt 9, such as pop-up TSRs, Microsoft Windows 3.1, IBM 5250 terminal emulation, 3270 Irma boards, and some implementations of BASIC.

Since Special Keyboard mode slows down the session, it should only be used when an application requires it. When in doubt, first try the default (OFF). If the application does not seem to respond, then set Special Kbd Handling to ON.

If you are using an international keyboard with an ALT-GR key to the right of the spacebar, set Special Kbd Handling to ON.

Precision Mouse **[ON] OFF**

Enables/disables Precision Mouse control, which greatly enhances remote mouse support. This feature will only be evident to the Control PC user. Precision Mouse displays a second mouse cursor on the Control PC's screen that shows the cursor's movement immediately, before the information is sent to the Host PC. This allows the Control PC user to accurately track the cursor's movement, even during lengthy screen updates. Precision Mouse is unnecessary during node-to-node sessions when throughput is fast enough for "real-time" mouse control. Applications which take direct control of the mouse may not support Precision Mouse control.

Inactivity T/O **0 - 999 [120]**

Specifies the number of seconds the Host PC will wait after no data has been sent or received before performing the action specified in

the Timeout Response field (RESTART or REBOOT). If this field is set to 0, the Host PC will not time out.

If this field is set to 0 and the DCD Loss Response field is set to IGNORE, the Host PC modem may reset itself immediately after carrier is lost, even though BHOST is not ready to process incoming calls. In this case, BHOST will not restart without manual intervention, but the modem will continue to answer calls. To restart BHOST manually from the Control PC, first connect to the Host PC's modem; then enter Terminal mode and type:

```
;DISC.
```

Note that you will not be able to see your keystrokes. This sequence will interrupt the BLAST protocol and allow BHOST to restart—it may also cause the Host PC's modem to hang up. After BHOST has restarted, you may log on as usual.

Timeout Response **[RESTART] REBOOT**

Specifies the action that the Host PC will take if an Inactivity Timeout occurs. RESTART prepares the Host PC for the next caller, disconnecting the current user. REBOOT forces the Host PC to perform a warm boot just as if it had been physically rebooted with the CTRL ALT DEL sequence.

NOTE: If this field is set to REBOOT, the Host PC will not necessarily reload BHOST—you must specify BHOST in the Host PC's AUTOEXEC.BAT file to insure that the Host PC will be ready to answer incoming calls.

DCD Loss Response **RESTART** **REBOOT [IGNORE]**

Specifies the Host PC's actions if the modem's Data Carrier Detect (DCD) signal is lost during a session.

RESTART – restarts BHOST after DCD loss and prepares for the next caller. This is the recommended setting if you are using a modem and have an appropriate connection between the system and modem.

REBOOT – reboots the Host PC after DCD loss. Note that, with this setting, BHOST will not necessarily be reloaded. If BHOST is not loaded from the Host PC's AUTOEXEC.BAT file, the Host PC will remain at the DOS prompt when rebooted.

IGNORE – ignores DCD loss. In order for BHOST to detect DCD Loss through an external modem, the modem cable must support the DCD signal. All standard modem cables support this signal.

IMPORTANT: If DCD Loss Response is set to IGNORE and carrier is lost during a session, the Host PC modem may reset itself immediately, even though BHOST is not ready to process incoming calls. In this case, BHOST will not restart and the Host PC will not be able to process incoming calls until the Logon T/O or Inactivity T/O takes effect.

Host Keyboard

[ON] OFF

Enables/disables the Host PC's keyboard. If this field is set to OFF when BHOST is started, the Host Keyboard is completely disabled; to regain control of the keyboard, you must reboot the Host PC or change this setting remotely. The Control PC may still initiate Chat Mode with the Host PC; in this case, the Host keyboard is enabled for the duration of the Chat.

IMPORTANT: If Host Keyboard is set to OFF and BHOST is started from the Host PC's AUTOEXEC.BAT, the Host PC's keyboard will remain disabled, even after rebooting. If this situation occurs, dial into the Host PC and change the Host Keyboard setting through SETBHOST.

This feature prevents unauthorized interference with a Control session.

Host Mouse

[ON] OFF

Enables/disables the Host PC's mouse. When this field is set to OFF, the Host mouse is *completely disabled*, preventing unauthorized interference with a session.

Host Screen

[ON] OFF

Enables/disables the Host PC's screen. When this field is set to OFF, the Host screen is completely disabled when BHOST is started, preventing anyone from seeing what is being sent to the Control PC's display.

When Host Screen is set to OFF, the Control PC may still initiate Chat Mode with the Host PC; in this case, the Host screen is enabled for the duration of the Chat.

IMPORTANT: If Host Screen is set to OFF and BHOST is started from the Host PC's AUTOEXEC.BAT, the Host PC's screen will remain disabled even after rebooting. If this situation occurs, try typing BHOST /k at the DOS prompt (you will not be able to see the characters on the screen). If that does not work, dial into the Host PC and change the Host Screen setting through SETBHOST.

Host Printer [NONE] LPT1 LPT2 LPT3

Specifies the Host PC printer to be used during a session. BHOST will monitor the printer port you specify here and redirect printing to the locations listed in the Printer(s) Enabled field. If you plan to print during a session, set this field to the Host PC's printer port. You may notice a slight performance decrease. If you do not plan to print during a session, set this field to NONE.

Printer(s) Enabled [NONE] CONTROL HOST BOTH

Specifies which printers will be active during a session. When an application issues a print command, the command will be executed on the printers specified here. Note that the Host Printer field must be set properly for this field to work.

NONE – printing is disabled.

CONTROL – enables only the Control PC's default printer.

HOST – enables only the Host PC's printer as specified in the Host Printer field.

BOTH – enables both Host and Control printers.

Session Command Window

If you are in Terminal mode, you may alter the BHOST session parameters via the Session Command window (Figure 18-4, next page). To open the Session Command window from Terminal mode, press ESC CTRL M. Commands are entered as lines of text using the following format:

parameter_command=value

where *parameter_command* is one of the parameter commands listed in the table below and *value* is a valid setting for the parameter (see preceding section for setting options).

To check the current value of a session parameter, simply type the *parameter_command* for the parameter. For example, to display the current value for the Host Keyboard parameter, type:

```
keyboard
```

To see the values for all session parameters, type:

```
settings
```

Each parameter will be listed along with its current value. The following commands are available:

<u>Parameter Command</u>	<u>Parameter</u>
DCDResp	DCD Loss Response
Inactimo	Inactivity T/O
Keyboard	Host Keyboard
Mouse	Host Mouse
Print	Printer(s) Enabled
PMouse	Precision Mouse
Printer	Host Printer
Screen	Host Screen
Scale	Scaling Ratio
Scan	Scan Interval
SKeyboard	Special KBD Mode
Sync	Sync Mode
TimoResp	Timeout Response

To close the Session Command window, press ESC.

```
C:\>
C:\>
C:\>
BLAST Host Session Command Mode
>settings

Current BLAST Host Settings

SCALE=1:1
SYNC=ON
SCAN=MEDIUM
PMOUSE=ON
SKEYBOARD=OFF
INACTIMO=120
TIMORESP=RESTART
DCDRESP=RESTART
MOUSE=ON
SCREEN=ON
KEYBOARD=ON
PRINTER=NONE
PRINT=NONE
>
^KH-help | DCD | CTS | | | 00:52:10 | | 81N | 12:47 PM
```

FIGURE 18-4

Appendix A

Error Messages

Introduction

The following is a list of BLAST error codes and a brief description of the cause of each error. Some error messages for other versions of BLAST are included in this list. Even though the codes may not apply to the version running on the local computer, they may occur on the remote system. Wording of messages may vary slightly depending on the version of BLAST.

BLAST Protocol Functions

20	loss of carrier during protocol logon
21	logon timeout (A BLAST protocol session was not established within the time specified by the BLAST protocol Logon Timeout. See Logon Timeout setup field on page 87 for details.)
22	console interrupt (The <i>ATTN</i> key was typed)
23	connect timeout

- 24 error in processing command file
- 25 cannot start protocol on remote system
- 26 Filetransfer terminated by remote system
(The remote system timed out during a BLAST protocol session or the remote operator pressed the *ATTN* key.)
- 27 attempt to connect with an incompatible private network
(There are special versions of BLAST that are limited to use within a particular network of systems. Use of these special versions outside of the network or use of a standard BLAST version within the network will give this message.)
- 29 connection control string timeout
- 30 loss of carrier during protocol connection

Transfer File Management

- 31 error-free file not found, or cannot be accessed
(Often occurs because the file or directory does not have read permission.)
- 32 error-free file cannot be created
(Often occurs because the file or directory does not have write permission.)
- 33 error-free file cannot be deleted
(Check permissions on the directory.)
- 34 error occurred while closing the error-free file
(This error occurs whenever BLAST cannot close an open file during Filetransfer mode.)
- 35 cannot position within the error-free file
(This error occurs when BLAST cannot close an open file during Filetransfer mode.)
- 36 error occurred while reading the error-free file
- 37 error occurred while writing to the error-free file
(Running out of disk space is a common cause of this error.)
- 38 size conflict
- 39 filename is too long or invalid
- 40 a file already exists with that name
- 41 error reading file directory
(Check the permissions of the directory.)
- 42 error writing to disk; disk is full
- 48 permission denied
(Your user profile on a multi-user system or the file attributes do not permit the current BLAST operation.)
- 49 transfer not allowed

Utility File Management

51	error opening a data file
52	error creating a data file
53	error deleting a data file
54	error closing a data file
55	error positioning within a data file
56	error reading from a data file
57	error writing to a data file
58	error in the size of a data file
59	error renaming a data file
60	BLASTDIR is invalid
61	SETUPDIR is invalid
62	OPTDIR is invalid

Scripting

65	script variable is READ-only; can't be set
66	user-defined script error command
67	cannot find entry in modems.scr or systems.scr
68	no matching label for GOTO
70	error executing command.com
71	all local commands complete
72	invalid switch specified
73	cannot overwrite or append
74	unknown file type
75	file already exists
76	too many open scripts
77	cannot load setup
78	setup already exists or cannot be created
79	not a valid directory
80	no setups found
81	no setup has been selected
82	upload cancelled
83	8-bit protocol requires an 8-bit channel; switching to 7-bit
84	packet size is too large; packet size too small for Access
85	remote control terminated by remote system
86	incompatible video mode
88	cannot initialize emulator
89	error printing, cannot open file
90	error processing a command file

Initialization

- 100 error allocating memory from the BLAST memory pool
- 101 environment variable TERM is too large
- 102 cannot extract control strings from terminal information database
(The TERM environment variable is not defined or the specified terminal type in TERM is incorrect.)
- 103 terminfo control string is too large
- 104 environment variable TERM is empty
(Set the TERM environment variable. Depending on operating system, you may have to “export” TERM.)
- 105 error allocating memory from the system
- 108 cannot load specified setup file
(The setup file specified does not exist in either the current directory or the directory specified by the SETUPDIR environment variable.)
- 109 error in processing translate table update file
- 110 usage error
- 111 cannot execute a child process
- 112 error creating a pipe
- 113 cannot fork
- 117 cannot ioctl () the console port
- 118 cannot open the console port
- 119 cannot ioctl () the communications port
- 120 cannot open the communications port
- 1) You may have selected an invalid communications port.
 - 2) Check the physical connection to the port. Make sure that the port specified is the actual port set up for communications.
 - 3) The port may be in use or may not have been released by another system process. Reboot the computer and load only BLAST to test the physical connection.
 - 4) The computer may be using an interrupt and/or base address that is not standard. Edit the BLAST.OPT to include proper address and IRQ.
 - 5) The hardware flow control (RTS/CTS) or Carrier Detect signals may not be configured to handle the port signals directly.
 - 6) Other applications may not have closed all ports when exiting. From the :\BLAST directory, type “blast /i” so that BLAST bypasses any checking of ports done by other applications.

121	a lock file exists for the communications port (Check the \usr\spool\uucp and/or \user\spool\locks directories for a LCK.Portname file. Delete the lock file if appropriate. This is a System Administrator function.)
122	error in terminal definition
123	function not available in background mode
127	control pipe io error
128	unexpected signal
133	network error occurred
134	BLASTNMP.EXE not loaded
135	network drivers not loaded (If using TCP/IP, be sure that the name of the TCP/IP TSR matches the one specified in BLAST.OPT.)
210	compression error
253	internal error

Script Processor

300–399	syntax error in command
400	too many strings

Network

502	fatal network error; BHOST terminated
-----	---------------------------------------

Appendix B

Key Definition Charts

BLAST Keys

Local BLAST functions are controlled by the following keys. They can be remapped with the BLAST Keyboard Utility, BLASTKBD (see “Keyboard Mapping Utility” on page 296). “EXT” indicates keys in the extended area of the keyboard, such as the numeric keypad.

<u>Function</u>	<u>PC Key 1</u>	<u>PC Key 2</u>
Cursor Up	UP (↑)	EXT UP
Cursor Down	DOWN (↓)	EXT DOWN
Cursor Left	LEFT (←)	EXT LEFT
Cursor Right	RIGHT (→)	EXT RIGHT
Home	HOME	EXT HOME
End	END	EXT END
Page Up	PGUP	EXT PGUP
Page Down	PGDN	EXT PGDN
Esc	ESC	
Del Char	DEL	EXT DEL
Insert Mode	INS	EXT INS
Help	F1	

Attention Key Sequences

Attention Key sequences are only active from Terminal mode. The sequences cannot be remapped, but the Attention key can be redefined by entering a new setting in the Attention Key setup field (page 72).

<i>ATTN</i> <i>ATTN</i>	Return to the Online menu.
<i>ATTN</i> <i>B</i>	Send a break signal (also interrupts an active BLAST script).
<i>ATTN</i> <i>C</i>	Toggle Capture mode on or off.
<i>ATTN</i> <i>E</i>	Start BLASTKBD, the BLAST keyboard remapping utility, with the current emulator selected.
<i>ATTN</i> <i>H</i>	Display Online Help.
<i>ATTN</i> <i>M</i>	Start BLASTKBD, the BLAST keyboard remapping utility.
<i>ATTN</i> <i>N</i>	Reset XON/XOFF Pacing.
<i>ATTN</i> <i>P</i>	Toggle printer logging on or off.
<i>ATTN</i> <i>R</i>	Reset the elapsed time clock.
<i>ATTN</i> <i>S</i>	Stop the elapsed time clock.
<i>ATTN</i> <i>0-9</i>	Start a BLAST Soft Key (digit is the Soft Key number).

Hot Keys

BLAST features Hot Keys for accessing certain functions from Terminal, Filetransfer, and Access modes. Not all functions are available from all modes (see chart below). Hot Keys are essentially macros that activate BLAST menu commands and return you to your starting point with just a few keystrokes. For example, in Terminal mode, typing `ALT F` starts Filetransfer mode and automatically returns you to Terminal mode when file transfer is completed.

Hot Keys are not available while BLAST scripts are running. To make Hot Keys active after an automated logon, be sure that the script command after `TERMINAL` is either `QUIT` or `RETURN`. You can remap Hot Keys with BLASTKBD, the BLAST keyboard remapping utility (see “Keyboard Mapping Utility” on page 296).

<u>Function</u>	<u>Key</u>	<u>Available Mode</u>
Background	ALT H	Always (if /b)
Abort BLAST	ALT X	Terminal
Connect	ALT C	Terminal
Disconnect	ALT D	Terminal
Capture	ALT O	Terminal
Learn	ALT R	Terminal
Select setup	ALT S	Terminal
Modify setup	ALT M	Terminal
New setup	ALT N	Terminal
Access	ALT A	Terminal
Local Edit	ALT E	Terminal, FT
Local Print	*	Terminal, FT
Local Type	*	Terminal, FT
Local List	ALT L	Terminal, FT
Local View	*	Terminal, FT, Access
Local System	*	Terminal, FT, Access
Filetransfer	ALT F	Terminal, Access
Remote Reboot	*	Access
Snapshot	*	Access
Chat	*	Access
Parameters	*	Access
Record	*	Access

FT=Filetransfer

* To avoid potential conflicts during remote control sessions, these keys do not have default values. When you assign keys through the BLAST keyboard mapping utility, BLASTKBD, remember that the values you pick will not be available to the Host PC programs during remote control sessions.

Terminal Emulation Keys

Press *ATTN E* or *ATTN M* from Terminal mode to view the BLASTKBD screen for these emulators (see “Terminal Emulation” on page 287).

DEC VT320 and VT220 Keys

<u>Function</u>	<u>PC Key</u>
Backspace	CTRL BACKSPACE
Del	BACKSPACE
Cursor Up	UP
Cursor Down	DOWN
Cursor Left	LEFT
Cursor Right	RIGHT
Keypad 0 – 9	KEYPAD 0 – 9
Keypad -	KEYPAD -
Keypad ,	KEYPAD *
Keypad Enter Key	KEYPAD +
Keypad .	KEYPAD .
PF1 – PF4	F1 – F4
Hold Screen	F5
Print Screen	ALT P
Toggle Auto Print	CTRL PRN
Scroll Left	CTRL LEFT
Scroll Right	CTRL RIGHT
Scroll Mode	(not mapped)
Find	INS
Ins Here	HOME
Remove	PGUP
Select	DEL
Prev Screen	END
Next Screen	PGDN
F6 – F12	F6 – F12
F13 – F20	ALT F3 – ALT F10
Help	ALT F5
Do	ALT F6
Shift 6 – F12	SHIFT F6 – F12
Shift F13 – F20	CTRL F3 – F10

DEC VT100 and VT52 Keys

Function

Backspace
Del
Cursor Up
Cursor Down
Cursor Left
Cursor Right
Keypad 0 – 9
Keypad -
Keypad ,
Keypad Enter Key
Keypad .
PF1 – PF4
Hold Screen
Print Screen
Toggle Auto Print
Scroll Left
Scroll Right
Scroll mode

PC Key

CTRL BACKSPACE
BACKSPACE
UP
DOWN
LEFT
RIGHT
KEYPAD 0 – 9
KEYPAD -
KEYPAD *
KEYPAD +
KEYPAD .
F1 – F4
F5
ALT P
CTRL PRTSC
CTRL LEFT
CTRL RIGHT
(not mapped)

ANSI Keys

Function

Backspace
Del
Cursor Up
Cursor Down
Cursor Left
Cursor Right
PF1 – PF4

PC Key

BACKSPACE
DEL
UP
DOWN
LEFT
RIGHT
F1 – F4

Data General D461, D411, D410, D200 Keys

<u>Function</u>	<u>PC Key</u>
Newline	ENTER
CR	CTRL ENTER
Break	ALT 255
Del	BACK
Erase EOL	KEYPAD -
Erase Page	DEL
Cursor Up	UP
Shift Up	CTRL UP
Cursor Down	DOWN
Shift Down	CTRL DOWN
Cursor Left	LEFT
Shift Left	CTRL LEFT
Cursor Right	RIGHT
Shift Right	CTRL RIGHT
Cursor Home	KEYPAD 5
Shift Home	CTRL KEYPAD 5
Print Window	KEYPAD *
Print Form	ALT P
Local Print	ALT T
SPCL	ALT Q
Hold	ALT Z
Send Graphics Cursor	ALT G
C1	HOME
C2	PGUP
C3	END
C4	PGDN
Shift C1	CTRL HOME
Shift C2	CTRL PGUP
Shift C3	CTRL END
Shift C4	CTRL PGDN
F1 – F12	F1– F12
F13 – F15	ALT 3 – 5
Shift F1 – F12	SHIFT F1 – F12
Shift F13 – F15	ALT 23 – 25
Ctrl F1 – F12	CTRL F1 – F12
Ctrl F13 – F15	ALT 13 – 15
Ctrl Shift F1 – F12	ALT F1 – F12
Ctrl Shift F13 – F15	ALT 33 – 35

WYSE60, WYSE50, TV920, D80, and ADM3A Keys

<u>Function</u>	<u>PC Key</u>
Backspace	CTRL BACK
Del	BACK
Enter	ENTER
Return	KEYPAD +
Back Tab	SHIFT TAB
Print	ALT P
Send	ALT B
Scroll Lock	ALT Z
Unlock Kybd	ALT U
Cursor Up	UP
Cursor Down	DOWN
Cursor Left	LEFT
Cursor Right	RIGHT
Home	HOME
Shift Home	CTRL HOME
Page Up	PGUP
Page Down	PGDN
Clear Line	END
Clear Screen	CTRL END
Del Char	DEL
Del Line	CTRL DEL
Ins Char	INS
Ins Line	CTRL INS
Ins/Replace	ALT I
F1 – F12	F1 – F12
F13 – F16	CTRL F3 – CTRL F6
Shift F1 – Shift F12	SHIFT F1 – SHIFT F12
Shift F13 – Shift F16	(not mapped)

HP2392 Keys

<u>Function</u>	<u>PC Key</u>
Soft Reset	ALT 1
Hard Reset	ALT 2
Backtab	SHIFT TAB
Return	ENTER
Enter	KEYPAD +
Home Up	HOME
Home Down	CTRL HOME
Cursor Up	UP
Cursor Down	DOWN
Cursor Left	LEFT
Cursor Right	RIGHT
Roll Text Down	CTRL DOWN
Roll Text Up	CTRL UP
Next Page	PGDN
Previous Page	PGUP
Insert Mode	INS
Delete Char	DEL
Insert Line	CTRL INS
Delete Line	CTRL DEL
Clear Display	END
Clearline	CTRL END
F1 – F8	F1 – F8
User Defined Keys	F9
User Sys Keys	F10
Default User Keys	SHIFT F10

IBM3101 Keys

<u>Function</u>	<u>PC Key</u>
PF1 – PF8	ALT F1 – F8
Cursor Home	HOME
Cursor Up	UP
Cursor Down	DOWN
Cursor Left	LEFT
Cursor Right	RIGHT
Ins Char	INS
Ins Line	KEYPAD 0
Del Char	DEL
Del Line	KEYPAD .
Erase EOS	F4
Clear	SHIFT F4
Erase EOF/EOL	F2
Erase Input	SHIFT F2
Print	F7
Print Msg	F5
Print Line	SHIFT F5
Aux	SHIFT F7
Send	F10
Send Msg	F6
Send Line	SHIFT F6
Newline	KEYPAD +
Back Tab	SHIFT TAB
Del Key	CTRL HOME
Reset	F9
Cancel	SHIFT F9
Prgm Mode	SHIFT F1
Attr Mode	F1
DS: Mode	ALT 1
DS: LTA	ALT 2
DS: Null Supp	ALT 3
DS: Auto NL	ALT 4
DS: Auto LF	ALT 5
DS: CR or CRLF	ALT 6
DS: Scroll	ALT 7

Appendix C

Troubleshooting

Problems in quotations are error messages.

Installing BLAST

“Invalid serial number”

The serial number consists of two capital letters followed by ten digits. Make sure that the number is being entered correctly. The serial number printed on the BHOST diskette will not be accepted as a BLAST serial number.

The 132-column mode isn't recognized.

Consult the documentation for your video adapter to find the mode number (a two-digit hexadecimal value) for 132-column support. Enter this number at the prompt.

Modem type isn't listed.

Use the “AT” modem type. Consult the modem's documentation for the form of flow control supported by the modem and choose setup parameters accordingly.

Starting BLAST

“Copy protection error”

If you receive this error on initial startup, reinstall the software, being careful to complete the entire installation—do not abort the process at any point.

“Can’t load setup”

Check SETUPDIR. Setup files should have the required extension .SU; make sure that the file you want to load has the correct name.

“Can’t open script file”

Check the path in the setup file or on the command line.

“Can’t find ... in MODEMS.SCR (or SYSTEMS.SCR)”

The BLAST support file has been modified but not re-indexed.

Going Online

“Can’t open the communications port”

Other communications software may be running on the same communications port that BLAST is attempting to use. Remove any TSRs (such as BHOST or resident fax software). Try starting BLAST with the /i switch (page 11).

“Illegal menu selection”

There is an error in the startup script. Check the setup for the name of the script, and examine the script for errors.

“Can’t connect to the remote system”

Check the setup for correct settings for Communications Port, Baud Rate, Data/Stop Bits, Parity, Modem Type, and System Type. If modem initialization strings are not seen on the screen, check cable and modem power.

“Modem is not responding”

Check the communications port setting in the setup, the modem cable, and modem power. For internal modems, make sure that a unique IRQ and base address has been specified for the port in

BLAST.OPT. Check that the modem is set to respond to “AT” commands with English language responses.

If you are running Windows, you can check your base address and IRQ number by doing the following:

- ◇ In Control Panel, double-click on System to open the System Properties window.
- ◇ Select the Device Manager tab.
- ◇ Locate and double-click on the commport you want to check.
- ◇ Click on the Resources tab.
- ◇ The first number in the “Input/Output Range” is the base address. The “Interrupt Request” number is the IRQ number.

If your modem is plug-and-play, you may not see a commport listing for it in Device Manager. If not, you may get your base address and IRQ number by doing the following:

- ◇ In Control Panel, double-click on Modems to access the Modem Properties window.
- ◇ Select the Diagnostic tab.
- ◇ Locate your modem in the list and highlight the commport opposite your modem type.
- ◇ Click on the “More Info” button to get the base address and IRQ number.

After getting your base address and IRQ number, make sure there are no conflicts and that your BLAST.OPT file reflects the correct settings. Remember that, if your IRQ number is higher than 9, you will need to give the hexadecimal equivalent in your BLAST.OPT file (see “BLAST.OPT Settings” on page 20).

If your modem is still not responding, you may have a Winmodem, which DOS BLAST does not support. Check with your modem documentation to find out if your modem is a Winmodem. If it is, you will need to use a different modem or upgrade your BLAST to our Windows product, Data Pump, which will work with Winmodems.

Network connection timeout.

Check network drivers for compatibility (see “BLAST Network Drivers” on page 28).

File Transfer

“Can’t log on with remote system”

Make sure you are using the correct protocol. If you are connecting to a multi-user host, such as a UNIX-based computer, you may have to start BLAST on the remote system in order to perform filetransfers. Consult the system administrator of the remote system for the proper command.

“Copy protection error”

If you receive this error during file transfer, it means that BLAST has encountered the same serial number on both the local and remote computers. It is illegal to run the same copy of BLAST on more than one computer.

Unable to perform remote commands

The remote system may have disabled access to remote commands. Consult the system operator of the remote system.

All filetransfer requests are ignored

Make sure that parity and data/stop bits settings match on both sides of the connection. Check that the 7-Bit Channel setup field in the BLAST protocol subwindow matches the setting on the remote system.

Too many retries

Make sure that flow control (XON/XOFF or RTS/CTS) is set correctly between each computer and its modem. If modem-based error control is available, make sure that it is being used.

Appendix D

The ASCII Character Set

D	H	O	M	D	H	O	M	D	H	O	M	D	H	O	M
0	00	00	nul	32	20	40	space	64	40	100	@	96	60	140	'
1	01	01	soh	33	21	41	!	65	41	101	A	97	61	141	a
2	02	02	stx	34	22	42	"	66	42	102	B	98	62	142	b
3	03	03	etx	35	23	43	#	67	43	103	C	99	63	143	c
4	04	04	eot	36	24	44	\$	68	44	104	D	100	64	144	d
5	05	05	enq	37	25	45	%	69	45	105	E	101	65	145	e
6	06	06	ack	38	26	46	&	70	46	106	F	102	66	146	f
7	07	07	bel	39	27	47	'	71	47	107	G	103	67	147	g
8	08	10	bs	40	28	50	(72	48	110	H	104	68	150	h
9	09	11	ht	41	29	51)	73	49	111	I	105	69	151	i
10	0A	12	lf	42	2A	52	*	74	4A	112	J	106	6A	152	j
11	0B	13	vt	43	2B	53	+	75	4B	113	K	107	6B	153	k
12	0C	14	ff	44	2C	54	,	76	4C	114	L	108	6C	154	l
13	0D	15	cr	45	2D	55	-	77	4D	115	M	109	6D	155	m
14	0E	16	so	46	2E	56	.	78	4E	116	N	110	6E	156	n
15	0F	17	si	47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20	dle	48	30	60	0	80	50	120	P	112	70	160	p
17	11	21	dc1	49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22	dc2	50	32	62	2	82	52	122	R	114	72	162	r
19	13	23	dc3	51	33	63	3	83	53	123	S	115	73	163	s
20	14	24	dc4	52	34	64	4	84	54	124	T	116	74	164	t
21	15	25	nak	53	35	65	5	85	55	125	U	117	75	165	u
22	16	26	syn	54	36	66	6	86	56	126	V	118	76	166	v
23	17	27	etb	55	37	67	7	87	57	127	W	119	77	167	w
24	18	30	can	56	38	70	8	88	58	130	X	120	78	170	x
25	19	31	em	57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32	sub	58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33	esc	59	3B	73	;	91	5B	133	[123	7B	173	{
28	1C	34	fs	60	3C	74	<	92	5C	134	\	124	7C	174	
29	1D	35	gs	61	3D	75	=	93	5D	135]	125	7D	175	}
30	1E	36	rs	62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37	us	63	3F	77	?	95	5F	137	-	127	7F	177	del

D – decimal; H – hexadecimal; O – octal; M – mnemonic

The chart below is a list of the standard ASCII control codes—with the decimal, hexadecimal, and octal values; the ASCII mnemonic; the key sequence, and a short explanation.

<u>D</u>	<u>H</u>	<u>O</u>	<u>M</u>	<u>Sequence</u>	<u>Explanation</u>
0	00	00	nul	<ctrl> @	used for padding
1	01	01	soh	<ctrl> A	start of header
2	02	02	stx	<ctrl> B	start of text
3	03	03	etx	<ctrl> C	end of text
4	04	04	eot	<ctrl> D	end of transmission
5	05	05	enq	<ctrl> E	enquire
6	06	06	ack	<ctrl> F	positive acknowledgement
7	07	07	bel	<ctrl> G	audible alarm
8	08	10	bs	<ctrl> H	backspace
9	09	11	ht	<ctrl> I	horizontal tab
10	0A	12	lf	<ctrl> J	line feed
11	0B	13	vt	<ctrl> K	vertical tab
12	0C	14	ff	<ctrl> L	form feed
13	0D	15	cr	<ctrl> M	carriage return
14	0E	16	so	<ctrl> N	shift out
15	0F	17	si	<ctrl> O	shift in
16	10	20	dle	<ctrl> P	data link escape
17	11	21	dcl	<ctrl> Q	device control 1 (resume output)
18	12	22	dc2	<ctrl> R	device control 2
19	13	23	dc3	<ctrl> S	device control 3 (pause output)
20	14	24	dc4	<ctrl> T	device control 4
21	15	25	nak	<ctrl> U	negative acknowledgement
22	16	26	syn	<ctrl> V	synchronization character
23	17	27	etb	<ctrl> W	end of text block
24	18	30	can	<ctrl> X	cancel
25	19	31	em	<ctrl> Y	end of medium
26	1A	32	sub	<ctrl> Z	substitute
27	1B	33	esc	<ctrl> [escape
28	1C	34	fs	<ctrl> \	frame separator
29	1D	35	gs	<ctrl>]	group separator
30	1E	36	rs	<ctrl> ^	record separator
31	1F	37	us	<ctrl> _	unit separator

D – decimal; H – hexadecimal; O – octal; M – mnemonic

Appendix E

Autopoll

The Autopoll Script

BLAST features Autopoll, a sample script that allows your unattended system to call a series of remote computers and exchange information. Autopoll performs the following tasks:

- ◇ reads a list of sites to be polled,
- ◇ connects to each site,
- ◇ executes a transfer command file to transfer files,
- ◇ disconnects,
- ◇ scans the log file to determine which transfers were successful,
- ◇ builds retry files as required,
- ◇ and adds the results to a status file.

Autopoll checks carefully for errors while polling. If an error is found, the problem site is scheduled to be retried. Only the file transfer commands that failed are attempted again.

Installing Autopoll

Autopoll consists of eight scripts that were copied into your BLAST directory when the BLAST program was installed on your system. The scripts are:

autopoll.scr – master script.
autoinit.scr – initializes variables and files.
autoierr.scr – reports initialization errors.
autodisp.scr – draws screen displays.
autoline.scr – reads site information.
autopsnd.scr – checks log for status of SENDs.
autoprcv.scr – checks log for status of GETs.
autoparx.scr – updates status files.

The scripts may be moved to any convenient directory in your system. For instance, you could segregate Autopoll from other BLAST files by creating a poll directory:

```
cd c:\blast
mkdir poll
move auto*.scr poll
```

In addition to these script files, you must have a BLAST setup called “autopoll” located in the BLAST Setup Directory. It must include a valid communications port and other connection information such as modem type and baud rate. You may also specify the script `autopoll.scr` in the Script File field of the setup, simplifying the command line to start Autopoll.

Starting Autopoll

Autopoll must be started from the directory in which the Autopoll scripts and support files (site and transfer command files) are found. If `BLAST.EXE` is not in this directory, you need to add the full path for `BLAST.EXE` to your `PATH` or give the full path in the command line. If `autopoll.scr` has been entered in the Script File field of the autopoll setup, the format for invoking Autopoll from the command line is:

```
blast autopoll max_cycles site_file [start_time]
```

If `autopoll.scr` has *not* been entered in the Script File field of the setup, the command line must explicitly include the script:

```
blast autopoll /sautopoll.scr max_cycles site_file [start_time]
```

The command line parameters have the following meaning:

`autopoll` the autopoll setup.

`/sautopoll.scr` the autopoll script.

`max_cycles` the maximum number of attempts to complete all specified transfers.

`site_file` the filename “stub” (the part of the filename before the extension) of the site description file.

`[start_time]` [optional] the time, in 24-hour format, that Autopoll will begin polling. The `WAIT UNTIL` command in `BLASTscript` requires the 24-hour format. If this parameter is omitted, Autopoll begins polling immediately.

`[TRACE]` [optional] the command to enable a capture file of the entire polling session. The capture file contains the text of login dialogs, modem initialization commands, and so forth. This feature is used primarily for troubleshooting.

Here are some example command lines:

```
blast autopoll 3 retail 10:45  
blast autopoll 2 daily 1:05 TRACE
```

In the first example, a maximum of three attempts will be made to poll the sites listed in the site file `retail.dat` starting at 10:45 am. Notice that the command line specifies just the stub “retail” of the site filename `retail.dat`. (Autopoll appends a variety of extensions to the filename stub to specify the names of special files.)

In the second example, a maximum of two attempts will be made to poll the sites listed in the site file `daily.dat` starting at 1:05 am, and a trace of the polling session will be made.

The Site File

The site file is the “master list” of information about the sites to be polled. Site files may use any valid filename, but the extension must be .dat. Each line in the site file holds the parameters needed to connect to and transfer files to and from one site. Each line, or site record, consists of five fields separated by exclamation marks, also called “bangs,” in the form:

```
setup_name!site_name!phone_number!baud_rate!TCF_name
```

where

<i>setup_name</i>	specifies a setup to be used for polling. If omitted, the field defaults to autopoll.
<i>site_name</i>	contains a descriptive label for the site. If omitted, the field defaults to the Description field of <i>setup_name</i> .
<i>phone_number</i>	specifies the phone number to be used for the site. If omitted, Autopoll uses the Phone Number field of <i>setup_name</i> .
<i>baud_rate</i>	specifies the baud rate to be used for this site. If omitted, Autopoll uses the Baud Rate field of <i>setup_name</i> .
<i>TCF_name</i>	specifies the transfer command file (TCF) to be used for this site. If omitted, this field defaults to autopoll.tcf.

Each line must contain four bangs. Any fields that are to be skipped must be indicated by consecutive bangs (!!). Blank lines and lines beginning with a space, tab, or pound sign (#) will be skipped, so you may freely comment your site file using these characters. Lines may not exceed 100 characters in length. Some example record lines are as follows:

[the ruler is shown to indicate column position]

```
1          10          20          30          40          50
|...|...|...|...|...|...|...|...|...|...|
!Blaster!1(919)542-0939!!
store06!!!!nightly.tcf
NewYork!Albany!782-8311!19.2!ny.tcf
```

In the first site record, no setup is specified, so `autopoll.su` will be loaded. The site name will be “Blaster,” overriding the Description field of the setup. The phone number will be 1(919)542-0939. The baud rate will be taken from the setup because that field is blank, and the transfer command file will default to `autopoll.tcf`.

In the second record, the setup `store06.su` will be loaded. The site name, phone number, and baud rate will default to the values given in `store06.su`. The transfer command file will be `nightly.tcf`.

In the last record, the file `NewYork.su` will be loaded. The site name will be “Albany,” the phone number will be 782-8311, the baud rate will be set to 19.2 kbps, and the transfer command file will be `ny.tcf`.

Transfer Command File

Autopoll uses a standard transfer command file (TCF) to specify files to be sent and received. You may use a unique TCF for each site listed in your site file, or you may use one TCF for multiple sites. For a complete description of the Transfer Command File, see “Transfer Command File” on page 117.

IMPORTANT: Autopoll treats wildcards and remote commands (such as remote print and remote rename) as “try one” specifications. These transfers and commands are attempted during the first cycle only. Even if errors occur, Autopoll does not attempt the transfers or commands again. For this reason, wildcards and remote commands should be used with caution.

Overview of Autopoll Script Actions

A brief overview of the basic actions of the autopoll scripts follows to give users a clearer understanding of the Autopoll process. Much of the error checking, which comprises most of the scripts, is not included.

1. `autopoll.scr` starts, reads the command line parameters, and puts them into variables.
2. If an error is found, `autopoll.scr` calls `autoierr.scr`, which reports errors and terminates the Autopoll session.

3. If no errors are found, `autopoll.scr` calls `autoinit.scr`, which initializes variables and files. Specifically, using the stub of the site file, `autoinit.scr` sets variables that allow `Autopoll` to create retry and summary files and to find stop and banner files (see “Other Files Using the Filename Stub” on page 351) to be used in the `Autopoll` session. `Autoinit.scr` then returns control to `autopoll.scr`.
4. `Autopoll.scr` calls `autoline.scr`, which reads and interprets the site file line by line for `@SYDESC`, `@PHONENO`, `@WORKTCF`, and `@LOGFILE` and returns control to `autopoll.scr`.
5. `Autopoll.scr` calls `autodisp.scr`, which displays on-screen status information during polling and then returns control to `autopoll.scr`.
6. `Autopoll.scr` uses variables gleaned from the site file by `autoline.scr` to begin file transfer of the first site. After it finishes the first filetransfer session, `autopoll.scr` loops back to call `autoline.scr` to get information for the next filetransfer session until it finishes attempting the complete cycle of file transfers.
7. `Autopoll.scr` calls `autoprcv.scr` and `autosnd.scr` to check the error-free log file for errors generated in the filetransfer sessions.
8. `Autopoll.scr` calls `autoparx.scr` to update the screen and status file.
9. If more than one cycle is designated in the command line, `autopoll.scr` uses the updated status file to retry any files that failed in the first cycle.
10. Steps 7–9 are repeated until all files have been successfully transferred or until the number of cycles designated in the command line has been completed.
11. `Autopoll.scr` quits

NOTE: `Autopoll.scr` also calls any userscripts that may be created. See the section “User-Supplied Scripts” on page 355 for details on creating these scripts and on the points at which `autopoll.scr` calls these scripts.

Configuration Example

Assume that you have been asked to set up a polling network for a client who has a central PC and two remote sites running BHOST. How do you set up Autopoll for this configuration? First, you install BLAST on the central and remote sites and verify that connections can be made reliably. This step is best performed interactively, that is, while you are present at the central system issuing commands directly to BLAST. When you are satisfied that BLAST is correctly installed, you need to create the following:

- ◇ setup files
- ◇ the site file
- ◇ transfer command files

The Setup Files

Suppose the sites are configured as follows:

<u>Site name</u>	<u>Phone</u>	<u>Login, password</u>
Sam's Discount Mart	542-0307	buz, apollo11
Metro Army Surplus	542-5694	neil, saturn5

Because the logins are different, different BLAST setup files are needed for each site. The setups, called “sam” and “metro,” are created by running BLAST at the central site (see “Creating a New Setup” on page 63).

The Site File

Using the setups, you could write a site file named retail.dat:

```
1           10           20           30           40           50
|...|...|...|...|...|...|...|...|...|...|
  Retail Site List for My Polling Network
sam!Sam's Discount!542-0307!!sams.tcf
metro!Metro Army Surplus!542-5694!!metro.tcf
```

The first line of the file is treated as a comment because it begins with a space. The last two lines are the actual site records. In this case, the site records may be duplicating information already specified in the Phone Number and Description fields of the setups. If so, the site records could be simplified:

```

1           10           20           30           40           50
|...|...|...|...|...|...|...|...|...|...|
Retail Site List for My Polling Network
(Phone number and Description loaded from setups)

```

```

sam!!!!sams.tcf
metro!!!!metro.tcf

```

The site file now has an additional comment line (five lines altogether); otherwise it is equivalent to the previous site list.

Transfer Command Files

According to the site list, a transfer command file called `sams.tcf` will be executed when Autopoll connects to Sam's Discount Mart, and the transfer command file `metro.tcf` will be executed when Autopoll connects to Metro Army Surplus.

Suppose you need to get two files from Sam and send one to him. The file `sams.tcf` might look like this:

```

1           10           20           30           40           50
|...|...|...|...|...|...|...|...|...|...|
+c:\buz\acq12.txt c:\client\sam1
+c:\buz\wk_82 c:\client\sam2
c:\tmp\message c:\tmp\read_me/OVW

```

As explained in “Transfer Command File” on page 117, the “+” sign in column 1 of a line signifies that BLAST will perform a GET. Thus, in the file `sams.tcf` above, BLAST will get `c:\buz\acq12.txt` and give it the local filename `usr\client\sam1`. BLAST will also get `c:\buz\wk_82` and give it the local filename `c:\client\sam2`. The absence of a “+” in the last line of the TCF signifies that BLAST will perform a SEND. Thus, BLAST will send `usr\tmp\message` and give it the filename `c:\tmp\read_me` on the remote system. The added `/OVW` switch signifies that BLAST will overwrite an existing file of the same name on the remote system (see “File Transfer Switches” on page 114 for more information about filetransfer switches).

`Metro.tcf` is similar to `sams.tcf`:

```

1           10           20           30           40           50
|...|...|...|...|...|...|...|...|...|...|
+c:\neil\acq12.txt c:\client\metro1
+c:\neil\wk_82 c:\client\metro2
c:\tmp\message c:\tmp\read_me/OVW

```

Where to Save Autopoll Files

The site file `retail.dat` and transfer command files `sams.tcf` and `metro.tcf` are created using the BLAST editor and *saved as text files only* in the same directory as the Autopoll scripts.

IMPORTANT: Autopoll script files, transfer command files, and site files must be stored in the same directory, which must be your current working directory.

Starting Autopoll

With the required files ready, the BLAST command line to start Autopoll might be:

```
blast autopoll 3 retail
```

which specifies a maximum of three attempts to complete the polling session with `retail.dat`.

Other Files Using the Filename Stub

Autopoll distinguishes several special files by appending different extensions to the site filename stub. The extensions for `retail.dat` are listed below (next page):

<u>Extension</u>	<u>Created by</u>	<u>Meaning</u>	<u>Example</u>
<code>.dat</code>	user (required)	Site file	<code>retail.dat</code>
<code>.stp</code>	user (optional)	Stop file	<code>retail.stp</code>
<code>.hdr</code>	user (optional)	Banner file	<code>retail.hdr</code>
<code>.log</code>	Autopoll	Short summary file	<code>retail.log</code>
<code>.prn</code>	Autopoll	Long summary file	<code>retail.prn</code>

Site File

The site file (`retail.dat`) is the master list of information about the sites to be polled.

Stop File

The stop file (`retail.stp`) is an optional file the user can create that allows BLAST to exit prematurely but gracefully from a polling session. Autopoll checks for the existence of the stop file in the Autopoll directory before each connection to a site. If the file is found, the polling session is terminated.

For example, suppose you want to halt Autopoll because you have found out that the files to be transferred to the last 10 sites of a polling session have been corrupted as a result of an error in database reporting. Creating a stop file—a file with the stub of the site file and the extension “.stp”—will allow BLAST to quit the polling session gracefully instead of connecting to the last 10 sites.

Since the existence of the stop file—and not its contents—signify to BLAST that a session should be terminated, the contents of the file are irrelevant. You can create a file called “retail.stp” using the BLAST editor. To ensure the completion of future transfers for the site file, Autopoll deletes the stop file before exiting.

Banner File

The banner file (retail.hdr) is an optional file created by the user. Autopoll prints the banner file prior to printing the summary file at the end of polling. Printing is performed by the BLASTscript LPRINT command. You might want this file to contain special text or graphics to distinguish the summary file within a large queue of printouts.

Long and Short Summary Files

Autopoll maintains two summary files, a long summary file and a short summary file. Prepared by Autopoll but not printed, the long summary file (retail.prn) is helpful for troubleshooting. Printed automatically at the end of polling, the short summary file (retail.log) is most helpful when polling goes well because a quick glance will confirm a successful polling session. The files are saved in the Autopoll directory.

A typical short summary file looks like this:

```
***** 02/09/96 11:15:29 *****
Cycle 1
  1. FAILED:Sam's Discount < Error transferring 3 file(s). >
  2. success: Metro Army Surplus

Cycle 2
  1. success: Sam's Discount
*****
Note: check retail.prn for complete session information.
```

A typical long summary file looks like this:

```
02/09/96 *****
11:15:33 * Cycle: 1   Site: 1
*
* Name: Sam's Discount
* Phone: 542-0307
* TCF: sams.tcf
* Log: ClS001.log
*
*----- SESSION INFORMATION -----
* Filetransfer error -8: DCD lost during transfer
* Error transferring 3 file(s). Log file follows:
*
* **** BLAST Professional UNIX 10.7.3 on remote system [uov]
* LOSS OF CARRIER, ending Filetransfer
* File transfer interrupted, 12% of file acql2.txt received
*****

02/09/96 *****
11:16:30 * Cycle: 1   Site: 2
*
* Name: Metro Army Surplus
* Phone: 542-5694
* TCF: metro.tcf
* Log: ClS002.log
*
*----- SESSION INFORMATION -----
* No errors encountered.
* Log file has been deleted.
*****

02/09/96 *****
11:18:49 * Cycle: 2   Site: 1
*
* Name: Sam's Discount
* Phone: 542-0307
* TCF: ClS001.tcf
* Log: C2S001.log
*
*----- SESSION INFORMATION -----
* No errors encountered.
* Log file has been deleted.
*****

02/09/96 *****
11:20:41 * Polling complete: all sites polled successfully.
*****
```

Tips and Tricks

Following are a few tips and tricks to help insure successful execution of Autopoll:

Keep it Simple

Polling sessions can quickly become complicated if several file transfers must be performed over a large network of remote sites.

Create simple but sensible directory structures to support the polling network. As a rule of thumb, command files should contain lines no longer than 80 characters so that they can be easily viewed and edited on standard terminals.

Go Step by Step

Build your network methodically. It may be worthwhile to set up only a few remote sites initially and use them to test the features of Autopoll. Add sites to the network in groups of five or ten, eliminating problems as you go, until the complete network is installed.

Problems Do Not “Just Go Away”

In a large polling network, it is not uncommon to have problems with a few remote sites; intermittent problems are especially frustrating. Take some time to examine these difficulties carefully because they can point to problems that actually affect the entire network. Following are some questions to ask in helping to identify a problem:

- ◇ Are the phone lines reliable?
- ◇ Could fax machines, answering machines, call waiting (or other phone company services) be interfering with modems making connections?
- ◇ Are the modems compatible with each other?
- ◇ Is BLAST or BHOST being initiated correctly on the remote?
- ◇ Are the expected files consistently present (on both sides)?
- ◇ Are directory and file permissions set appropriately?

Tune BLAST Protocol Parameters

Some BLAST protocol parameters, such as the following, can be tuned for better performance with Autopoll:

```
Logon Timeout:      20
Inactivity Timeout: 20
DCD Loss Response: ABORT
```

These settings noted above permit Autopoll to react more quickly to lost connections than do the default settings. You may also wish to experiment with compression levels and packet size to find settings for best throughput. If your remote sites are running BHOST, bear in mind that the highest compression level supported by BHOST is

1 unless additional memory is allocated for compression buffers. Consult the *BHOST User Manual* for further information.

Modifying Autopoll

Because Autopoll is written in BLAST's scripting language, it is easy to customize and is thoroughly commented.

User-Supplied Scripts

The behavior of Autopoll can also be changed by writing one or more user-supplied scripts. Because Autopoll checks for the existence of these scripts at various points during execution, the scripts should be named as shown below. If Autopoll finds a user-supplied script, the script is executed by the BLASTscript CALL command. Autopoll tests the value of @STATUS when the called script returns command to Autopoll; polling continues normally if @STATUS equals 0; otherwise the site is marked as failed.

User-supplied scripts reside in the same directory as the Autopoll scripts. They are called at the following points during execution:

autousr0.scr	before the first site is polled (polling is aborted if this script fails).
autousr1.scr	before every attempt to CONNECT.
autousr2.scr	before every attempt to start FILETRANSFER.
autousr3.scr	before every attempt to DISCONNECT.
autousr4.scr	before Autopoll terminates.

Because BLASTscript variables are global, a user-supplied script must not disturb the contents of any variables needed by Autopoll. The following variables may be changed freely by any user-supplied script:

@STATUS	@EFERROR
@input	@temp
@xferok	@msg
@start	@filename

You can also create new variables if you wish. To help prevent confusion, begin new variables with “u”, for example, @uvar2.

File I/O with User-Supplied Scripts

Autopoll opens files specified by file handles 1 through 7 at various points during execution. The handles have the following functions:

1	read-only	current site (or retry) file.
2		utility I/O.
3		utility I/O.
4		utility I/O.
5	write-only	complete polling results.
6	write-only	retry file for next cycle.
7	write-only	brief polling results (printed out).

Any of the handles reserved for utility I/O may be opened by user-supplied scripts as long as the handles are freed before the scripts return to Autopoll (i.e., each user script must close its own files). User scripts may also write to the status files specified by handles 5 and 7

Autopoll closes the standard BLAST log file before calling user-supplied scripts. If a user script opens a log of its own, the log must be closed before execution returns to Autopoll.

Sample User-Supplied Script

The following user-supplied script allows a user to create a TCF file from a directory listing, thereby avoiding the need to know specific filenames or to use wildcards, which prevent filetransfer retries.

```
# autousr0.scr
#
# This script extracts the names of text files from a directory and uses the
# names to create a tcf file. Without the user having to know specific
# filenames, the script creates a tcf file that will transfer to the remote
# system all the text files from a specific directory, thus avoiding the use
# of wildcards, which prohibits filetransfer retries.
#
# The script returns the following errors:
#
# 0 - no error
# 1 - cannot delete file
# 2 - cannot create/open file
# 3 - no files found in directory
#
if exist "files.log" ldelete "files.log
if @status not = "0" return 1
set @logfile = "files.log"
l!list long "c:\\stinven"           # capture dir listing in log file
set @logfile = ""
```



```

if exist "auto.log" ldelete "auto.log"
if @status not = "0" return 1
set @logfile = "auto.log"
set @ustring1 = ".txt" # set strings to find filenames
set @ustring2 = "*"
let @ucount = "0"
fopenr 2, "files.log"
if @status not = "0" return 2
if exist "invent.tcf" ldelete "invent.tcf"
if @status not = "0"
    fclose 2
    return 1
end
fopena 3, "invent.tcf" # create tcf file to writing to
if @status not = "0"
    fclose 2
    return 2
end
.SEARCHLOOP
fread 2, @uline # read log file & extract filenames
if @status = "0"
    strinx @uline, @ustring1
    if @status not = "0"
        let @ucount = @ucount + "1"
        let @uposend = @status + "3"
        strinx @uline, @ustring2
        let @upostart = @status + "8"
        set @ufname = @uline
        strtrim @ufname, @upostart, @uposend
        fwrite 3, "c:\\stinven\\", @ufname, " c:\\newinv\\", @ufname # write file-
        # names to tcf
    end
    goto .SEARCHLOOP
end
fclose 2
fclose 3
ltype "invent.tcf" # type tcf file for testing
if @ucount = "0" return 3 # no files found - will be aborted
set @logfile = ""
return

```

This script can be modified to create more than one TCF and to create TCFs for the remote system

Configuration Worksheets

The following worksheets may help you organize the large amount of information needed to set up a polling network successfully.

A. List Machines

List the machines in your polling network. For completeness, include information for the central site as well.

Site Name Phone Modem Type Port BLAST Version System Type

Central

- 1.
- 2.
- 3.

B. Decide on Setups

Decide whether or not different setup files will be needed for each site. If so, create the setups and list their names. Remember, Auto-poll loads the setup autopoll.su by default.

Site Name Setup Name

- 1.
- 2.
- 3.

C. Set Up the Remote Sites

Set up the remote sites and test each connection manually. Make sure the following sequence of keyboard commands work flawlessly:

Connect	dials the modem and logs in if necessary.
Filetransfer	enters BLAST filetransfer.
ESC	exits BLAST filetransfer.
Disconnect	logs off and hangs up the phone.

D. Create the Site File

Build the entries in the site file with any standard text editor, selecting appropriate name(s) for the TCF files.

site filename: _____ .dat

Setup Name Phone Baud TCF

E. Create the Transfer Command Files

List the files to be transferred to and from each site and the direction of transfer (S=SEND, G=GET). Afterward, write the various TCF files and put them in the autopoll directory.

<u>Site</u>	<u>S/G</u>	<u>Remote Name</u>	<u>Local Name</u>	<u>Options</u>
1.				
2.				
3.				

F. Decide on Cycles

Decide how many cycles to allow for polling and when to start:

Cycles:

Start time:

G. Build the Command Line to Start Autopoll

Use the following format:

```
blast autopoll /sautopoll max_cycles site_file [start_time]
```

H. Check Environment Variables

Check the values of BLASTDIR, SETUPDIR, and PATH. When they are correct, change to the autopoll directory, type in the command line, and let Autopoll take over!

Appendix F

Glossary

Acknowledgement

Confirmation by the receiving computer to the sender that the transmission of packets is complete and accurate. With BLAST protocol, the frequency at which acknowledgements are requested may be set by the user via the ACK Request Frequency setup field (page 90) and the reserved variable @ACKFREQ (page 240).

Asynchronous

Refers to the mode of data communication that provides a variable time interval between characters during transmission. The data is sent at irregular intervals, but each character is preceded by one start bit and followed by one stop bit. This is the opposite of synchronous data transmission in which characters are sent at a fixed time interval.

BLASTDIR

The disk directory where BLASTscripts and Online Help files are stored. BLAST records this directory as part of its installation process. You may change your BLASTDIR by running BINSTALL with the CUSTOM option, by making a BLASTDIR assignment in

your BLAST.OPT file (page 21), or by setting the BLASTDIR environment variable to the value of the new directory (page 8).

BLAST Keys

Keys used for program control, such as CTRL K H for online help (see “BLAST Keys” on page 298).

BLAST.OPT

The BLAST options file that allows you to customize BLAST’s settings. See “BLAST.OPT” on page 19 for more information.

BLAST Protocol

Also called BLAST session protocol, BLAST’s, Inc., proprietary sliding-window protocol. BLAST protocol is designed to work over 7-bit and 8-bit connections and is compatible with both software and hardware flow control. Other features include: simultaneous bi-directional transfer, adjustable packet and window size, restart from the point of interruption, data compression, text translation between systems, and remote file management. For more information on BLAST protocol, see Chapter 6.

BLASTscript Programming

BLAST’s built-in programming language designed for communication applications. BLASTscript allows you to automate partially or totally any communications task you would otherwise do interactively. For more information on BLASTscript, see Chapter 12 (*Introduction To Scripting*) and Chapter 13 (*BLASTscript Topics*).

Compression

Automatic and systematic reduction of the number of characters sent during BLAST protocol transfer. By applying a set of compression rules to the data stream at both ends of a communications link, data can be reconstructed intact. For more information, see “Compression Levels” on page 121.

CRC-16

An acronym for Cyclic Redundancy Check, which is a technique for detecting data packets that have been altered by noise. This tech-

nique, used by BLAST, is the one used in IBM SNA/SDLC networks and in X.25 packet-switching networks.

Current Menu

The currently operative BLAST command menu. The possible values are Offline, Online, Filetransfer, Access, Local, and Remote.

Data Stream

The flow of characters between two computers over a communications link. The stream is serial (that is, one bit after another).

Editor

The built-in text editor BLASTEDT. Chapter 11 fully describes the use of the Editor. You may also link your own text editor for use in BLAST by including an EDITOR assignment in BLAST.OPT (see the discussion of EDITOR on page 24 for more information).

Emulator

A program that makes a standard display appear as if it is a particular terminal type to another computer. The emulation ensures that the keyboard and display simulate those of the chosen terminal as closely as possible. BLAST offers several terminal emulators (see “Terminal Emulation” on page 287).

Flow Control

Regulation of the flow of information between computers so that no data is lost. There are two types of flow control, one-ended and two ended:

One-ended flow control is performed entirely by one computer and can be used only for text uploading. The other system does not participate in the flow control process. Examples of one-ended flow control are character delay (in which each character transmitted is followed by a pause) and line delay (each line transmitted is followed by a pause).

Two-ended flow control is more common because it involves both computers working together to avoid data loss while maintaining data stream movement as rapidly as possible.

FTP

Standard protocol for file transmission over TCP networks. FTP (an acronym for file transfer protocol) was designed to promote sharing of files across networks while shielding users from variations in how files are accessed and stored on different computer systems. For more information on BLAST's implementation of FTP, see Chapter 7.

Full-Duplex

A telephone term that refers to the ability of a circuit to carry sound in both directions at the same time. Almost all telephone connections are full-duplex, that is, each party may hear and speak simultaneously.

In data communications, full-duplex refers to simultaneous transmission to and from a remote computer. See also "simultaneous bidirectional transfer" below.

Half-Duplex

A telephone term that refers to the inability of a circuit to send and receive sound and data in both directions at the same time. For both ends of a half-duplex circuit to communicate, they must establish a protocol to intermittently "turn the line around."

In data communications, this term refers to transmission of data in only one direction at a time.

Host Mode

The mode of operation of a multi-user computer in which a remote computer calls the multi-user computer, logs onto it, and starts a communications session. The remote machine has control of the multi-user computer in the transfer.

Interactive

Control of a program via the keyboard by pressing keys to make commands happen. This differs from performing the same function via a script.

Kermit

Protocol created by Columbia University for file transfer. Some versions of Kermit can act as host “servers,” allowing other computers to dial in and control a Kermit session. For a full discussion of Kermit and how it is may be implemented by BLAST, see Chapter 8.

Keyboard Mapping

Process of selecting which physical keys on your keyboard will send commands and data to the remote computer. If a key sequence is awkward, or a key does not exist on your PC, BLAST allows you to “map” that function to another keystroke using the BLASTKBD utility. Mapping allows you to create custom keyboards and emulators for many tasks (see “Keyboard Mapping Utility” on page 296).

Launch String

String that signals to the network interface unit where to break the BLAST protocol data stream into packets. Launch string is especially important for sessions using protocol converters. The default launch string for BLAST is the ASCII carriage return code. This string may be reassigned using the Launch String setup field (page 91) or the reserved variable @LAUNCHST (page 256).

Local Computer

Computer in front of you, which you operate via its keyboard.

Menus

Displays containing commands and functions from which the user may chose.

Modems

Electronic devices that transfer data over telephone lines from one computer to another.

MODEMS.SCR

Portion of BLAST’s script library containing controlling statements for the modems available in the Modem Type setup field. Connect and Disconnect commands from the Online menu invoke

MODEMS.SCR in order to get modem information for connecting and disconnecting from a remote computer.

Online Help

Context-sensitive information available about BLAST and its use. For help on a specific menu command or setup field, place the cursor on that item and press F1. While connected as a terminal to another system, press CTRL H for Online Help.

OPTDIR

Environment variable that specifies the directory where BLAST and BHOST look for the BLAST.OPT file, which is normally located in BLASTDIR. By setting OPTDIR to a directory containing an alternative BLAST.OPT file, you may temporarily override existing BLAST.OPT settings. For example, network users may place a separate BLAST.OPT file in a directory other than BLASTDIR (for more on the use of OPTDIR in configuring LANs, see page 17).

Packet

The unit in which BLAST protocol transmits data. The amount of data in each packet may be set by the Packet Size setup field (page 99) and @PAKTSZ reserved variable (page 259) in order to optimize data throughput over various line conditions. Over a noisy telephone line, smaller packets are more likely to get through unaltered. For lines not normally subject to noise, such as a direct-connect cable, larger packets can maximize throughput.

Padding

Extra characters added to the beginning or the end of a file. Padding is often required when transferring fixed-length records to and from a remote computer, or when a data file must contain only certain characters as nulls.

Propagation Delay

The length of time from when a character is transmitted to the time it is received. A propagation delay of as little as a half-second can dramatically degrade throughput for some communications applications. BLAST, however, is highly resistant to propagation delay.

Protocol

A set of rules controlling the method by which data is transmitted and received. Both computers in a data exchange must be using the same protocol or the data will not be sent and received correctly.

Remote Computer

Computer with which your local computer is communicating during a transfer session.

Retransmission

Transmission of a packet of data that has already been sent but has been corrupted in the original transmission (usually because of telephone line noise). With BLAST, the sending computer retransmits a packet until it arrives intact at the receiving machine.

Scrolling Region

Area of the screen displaying program status and data when operations are being performed.

Setup

A file containing all the information necessary for BLAST to communicate with another system, including terminal parameters such as emulation and parity. The user may create, change, and delete setups from within BLAST. All Setup files end with a “.SU” extension. For detailed information on setups, see Chapter 5.

SETUPDIR

The disk directory where setups are stored. BLAST records this directory as part of its installation process. You may change your SETUPDIR by running BINSTALL with the CUSTOM option, by making a SETUPDIR assignment in BLAST.OPT (page 26), or by setting the SETUPDIR environment variable to the value of the new directory (page 9).

Simultaneous Bi-Directional Transfer

Refers to the ability to transfer files both to and from a computer simultaneously. Unlike other file transfer applications, BLAST supports simultaneous bi-directional transfer.

Sliding Window

Design that allows the transmission of data packets while the computer is waiting for acknowledgements that previous packets were correctly received. This design, employed by BLAST, makes transfer much more efficient over circuits with long propagation delays, such as satellite links and network virtual circuits. For a detailed discussion of sliding window, see “Bi-Directional and Sliding-Window Capability” on page 103.

SoftKeys

A set of ten single keystrokes that send often-used character strings to the remote system. The character strings are mapped by the user. See “Soft Keys” on page 298 for details.

SYSTEMS.SCR

Portion of BLAST’s script library containing identifying characteristics of the multi-user systems available in the System Type setup field. Filetransfer and Upload commands from the Online menu invoke SYSTEMS.SCR in order to get system information to carry out file transfer and uploading.

Terminal Definition File

Holds the values of the BLAST Keys and the terminal characteristics of the local video display. This file, called BLAST.TDF, may be modified using BLAST’s keyboard mapping utility, BLASTKBD (see “BLAST Keys” on page 298).

Text Capture

Direct transfer of incoming text from a remote computer. For more information, see “Downloading Text from a Remote Computer” on page 146.

Text Upload

Direct transfer of text to a remote computer. For more information, see “Uploading Text to a Remote Computer” on page 145.

Throughput

The measurement of the total amount of data transferred between two computers. Throughput is usually measured in characters-per-second, which is ten times the bits-per-second.

TMP

Environment variable that specifies the directory where BLAST and BHOST save temporary files. Setting TMP to a RAM drive can significantly enhance performance. For more information, see the discussion of TMP on page 9.

Translation

The filtering of selected characters in the data stream while in Terminal mode and during text uploading and downloading. Via a translate file specified in the Translate File setup field, a user may substitute one character for another. You may also completely remove a character from an uploaded or downloaded file. For more information on translation, see “Data Stream Filtering and Alteration” on page 283.

Video Suppression

The mode of operation in which an application, such as BLAST, does not read or write to the video display.

Xmodem

Half-duplex protocol that uses 128-byte packets and works only on 8-bit systems. Because it cannot transmit the name of the file, Xmodem requires that the filename be entered on the receiving system. Also, because it can only transmit 128-byte packets, Xmodem pads the last packet to 128 bytes, thus often increasing the size of the original file. For more information on BLAST’s implementation of Xmodem, see “Xmodem Protocol” on page 140.

Ymodem

Half-duplex protocol that uses 1024-byte packets. Unlike Xmodem, it transmits the filename as part of the file transfer and can transmit the exact file size. For more information on BLAST’s implementation of Ymodem, see “Ymodem Protocol” on page 141.

Ymodem G

A variation of Ymodem that transmits a continuous stream of packets without error correction. If an error occurs, the receiver sends an error signal that causes the entire file transfer to abort.

Zmodem

A sliding-window protocol that uses variable packet sizes and works over both 7- and 8-bit connections. It is compatible with both software and hardware flow control and can restart a file transmission from the point of interruption. For more information on BLAST's implementation of Zmodem, see "Zmodem Protocol" on page 142.

Index

Symbols

/APP

- BLAST Protocol 114
- Kermit Protocol 134

/COMP=*n* 114

/FWD

- BLAST Protocol 114
- Enabling/Disabling 91, 123, 248

/OVW

- BLAST Protocol 114
- Enabling/Disabling 91, 248

/STR

- BLAST Protocol 114–115
- Enabling/Disabling 91, 123, 248

/TXT

- BLAST Protocol 115

@STATUS 267–268

- Commands Set by 204–205
- Saving Value of 167–168

A

Access Menu 308–311

- Chat 309
- Parameters 309
- Record 310
- Snapshot 310

Access Mode 306–311

- File Transfer from 307
- Hot Keys 310–311
- See also* Access Menu

ASCII

- Character Set 341
- Control Codes 342
- Script Command 205

Attention Key. *See* ATTN Key

ATTN Key 41, 50–51

- Aborting Scripts 156
- Sequences 328
- Setup Field 72

Automation

- BLAST Protocol 121
- Scripting 59–60

See also Autopoll

Autopoll 343–359

- Banner File 352
- Command Line 344–345
- Configuration 349–351, 357–359
- Installing 344
- Modifying 355–357
- Remote Commands 347
- Setup 344–345, 346, 349
- Site File 346–347, 349–350
- Starting 344–345, 351
- Stop File 351–352
- Summary Files 352–353
- Tips 353–355
- Transfer Command Files 347, 350
- User-Supplied Scripts 355–357
- Wildcards 347

B

Background Mode 13–19

- Adjusting BLAST.OPT Settings 15–19
- Hot Key 13, 249
- LANs and 17–18
- Limitations 14–15
- Memory Use 15–19
- Starting 14

Baud Rate

- Autopoll Site File 346
- Compression Level and 122
- Reserved Variable 242
- Setup Field 70
- TTY Emulation 290
- Zmodem Block Length and 97–98, 279

BHOST 303–319

- BLAST Setup 305
- Compression Level 305
- Login 305
- Modifying Settings 312–319
- Online Menu Options 306–307
- Packet Size 100, 305
- Security Features 304
- Session Parameters Window 313–318
- Starting BLAST 106–107
- Transferring Files 307
- See also* BHOST Settings

BHOST Settings 312–319

- DCD Loss Response 316–317
- Host Keyboard 317
- Host Mouse 317
- Host Printer 318
- Host Screen 317–318
- Inactivity T/O 315–316
- Precision Mouse 315
- Printer(s) Enabled 318
- Scaling Ratio 313–314
- Scan Interval 314–315
- Special KBD Handling 315
- Sync Mode 315
- Timeout Response 316
- BLAST**
 - Background Mode 13–19
 - Host Mode 303–319
 - On a LAN 28–35
 - Other Programs and 176–177
 - Removing from Memory 15
 - Screen 38–40
 - Starting 37–38
 - Unattended 343–359
- BLAST Editor** 149–153
 - Cursor Movement 151
 - Flow Control and 149
 - Inserting and Deleting Text 151
 - LANs and 35, 149–150
 - Managing Text Blocks 152
 - Quick Reference 150–151
 - Quitting 153
 - Scrolling 151
 - Searching 152–153
- BLAST Key Set** 296–300
 - BLAST Keys 296, 298–299
 - Hot Keys 296, 299–300
 - Soft Keys 296, 298
- BLAST Keys** 296, 298–299
 - Definition Charts 327–329
 - Frequently Used 50–51
- BLAST Protocol** 101–124
 - Advantages 102–103
 - Automating 121
 - Circuit Requirements 104
 - Compression Level 121–122
 - CRC Error Detection 103
 - Design 103–104
 - Ending File Transfer 108–109
 - Extended Logging 278–279
 - File Transfer 110–124, 178–181
 - File Transfer Switches 114–115
 - File Transfer Templates 113
 - Filetransfer Menu 56–57, 110
 - Fine-Tuning 121–122
 - Getting Files 112, 178–179
 - Message 56, 179, 213–214
 - Packet Acknowledgement 90, 103–104, 240
 - Packet Size 104, 121, 259
 - Remote Commands 179, 214–215
 - Remote Menu 110, 120–121
 - Restarting Interrupted Transfer 116
 - Scripting Considerations 180–181
 - Security 123–124
 - Sending Files 111–112, 178–179
 - Setup Subwindow 87–92
 - Slave Mode 198–199
 - Starting File Transfer 105–108
 - Timeouts 107–108
 - See also main entry* Timeouts
 - Transfer Command Files 117–120, 179
 - Transfer Options 110–111
 - Transfer Password 88–89, 123–124, 269
 - Transfer to a Printer 116–117
 - Wildcards 112–113
 - Window Size 89, 274
- BLAST Session Protocol.** *See* BLAST Protocol
- BLAST.OPT** 19–28
 - Background Mode and 19
 - Command Format 19–20
 - LAN Drivers and 19
 - Multiple Files 19
 - Search Path 19
 - See also* BLAST.OPT Settings
- BLAST.OPT Settings**
 - 16550 20
 - ACTFILE 20
 - Adjusting for Background Mode 15–19
 - BANNERTIME 20
 - BG_BLK_SIZ 16, 17, 20–21
 - BLASTDIR 21
 - BREAKLEN 21
 - COMMPORT 21–23
 - COMPBUF 23

- EDITOR 24
- FILEBUF 16, 17, 24
- GROUND 24–25
- LOGFILE 25
- MEMPOOL 16, 17, 25
- NETSERVICE 25
- PRINTERCHK 25–26
- SETUPDIR 26
- SLICE 16, 26
- SPEAKER 26
- TCPINGWLAD 27
- TCPININLOCAD 27
- TCPINSNMSK 27
- TTRAP LEN 27
- VIDEOBUF 17, 27
- VIDEOMODE 28
- BLASTAT.EXE 15
- BLASTDIR
 - BLAST.OPT Setting 21
 - Environment Variable 8
- Blaster (Online Demonstration and Testing Service) 42–47
 - Connecting to 44–45
 - File Transfer 45–47
 - Logging Off 47
 - Setup 43
- BLASTKBD 296–302
- BLASTscript. *See* Script Commands and Scripting
- Buffers
 - Data Compression 23
 - FIFO 20
 - File 24

C

- CALL Statement 168–169, 206–207
- CANCEL Key 51
- Capture 55, 147, 233–234
- Chat, Access Menu Option 309
- Command Area 38–39
- Command Line Arguments 10–11
 - Autopoll 344–345
 - Passing Information to BLAST 177
- Command Line Switches 10–13
 - /b 11
 - /c 11

- /i 11
- /n 11, 175, 252, 265
- /px 11
- /q 11
- /sscriptname 10, 156, 177
- /tx 12
- /v 12
- /w 12
- /x 12, 278–279
- /y 12
- argument 10–11, 177, 344–345
- Autopoll 344–345
- For Loading Drivers 29–32
- setupname 10
- Xmodem Protocol 140
- Ymodem Protocol 140
- Communications Port
 - BLAST.OPT Setting 21–23
 - Reserved Variable 242
 - Setup Field 67–68
- Compression Buffers 23
- Compression Level
 - BHOST 305
 - BLAST Protocol 121–122
 - Reserved Variables 243, 261, 264
 - Setup Fields 91–92
- CONNECT Statement 159, 208
- Connecting 159, 193–196
 - Troubleshooting 338–340
- Connection Timeout 68
- CRC Error Detection 103

D

- Data Stream
 - Alteration 283–287
 - Control 283–302
 - Filtering 283–287
 - Substitution 286–287
 - Translate File 284–287
 - Translation 286
- Date Format
 - @DATE 244
 - @DATEFORMAT 244
 - @LOGDATEFORMAT 256
- DCD Loss Response 89, 180–181, 316–317
- Default

- Setup 63
- Values for Reserved Variables 240
- Demo Line. *See* Blaster
- Demonstration Service. *See* Blaster
- Dialing Directory 62
- Disconnecting 193–194, 196–198
- DISPLAY Statement 176, 209
- Documentation 3–5
- Downloading Text 146–147, 192
- Driver Command Line Switches
 - BLASTACS 29–30
 - BLASTIPX 31–32
 - BLASTLAN 32–33
 - BLASTNMP 30–31
 - BLASTTCP 33

E

- Echo
 - Local 84, 256
 - Password Security and 260
 - Script Command 209–210
 - Wait for 85, 275
- Editor
 - BLAST.OPT Setting 24
 - Executing from Background Mode 14–15
 - Executing from Menu 58
 - See also* BLAST Editor
- Emulation. *See* Terminal Emulation
- Emulator Maps 296, 300
- Environment Variables 8–9
- Error Checking 188–189
- Error Detection, CRC 98, 103, 280
- Error Messages 321–325
 - BLAST Protocol Functions 321–322
 - Initialization 324–325
 - Network 325
 - Script Processor 325
 - Scripting 323
 - Transfer File Management 322
 - Utility File Management 323
- Extended Logging 278–279

F

- File Transfer
 - BHOST 307
 - BLAST Protocol 110–124, 178–181

- Blaster, with 45–47
- Error Checking 188–189
- FTP 126–129, 181
- Kermit Protocol 131–135, 181–184
- Troubleshooting 340
- Xmodem Protocol 140–141, 184–185
- Ymodem Protocol 141–142, 185–187
- Zmodem Protocol 142–143, 187–188
- File Transfer Status Area 39–40
- File Transfer Switches
 - BLAST Protocol 114–115
 - Kermit Protocol 134
 - Security with 123
 - Setup Fields for Enabling 91
 - See also* Filename Restrictions and specific file transfer switches
- File Transfer Templates 113
- Filename Restrictions
 - BLAST Protocol 115–116
 - FTP 128–129
 - Kermit Protocol 135
 - X, Y, and Zmodem Protocols 143–144
- Filetransfer Menu 56–57
 - BLAST Protocol 110
 - FTP 126–127
 - Kermit Protocol 131–132
 - Xmodem Protocol 56
 - Ymodem Protocol 56
 - Zmodem Protocol 56
- FILETRANSFER Statement 160, 211–216
 - See also* File Transfer
- Filtering
 - Data Stream 283–287
 - VT Sequences 90, 248
- Flow Control 35–36
 - Downloading Text 146–147
 - Editor and 149
 - Uploading Text 145
- FTP 125–130
 - Commands 129–130
 - Ending a Session 129
 - File Transfer 126–129, 181
 - Filename Restrictions 128–129
 - Filetransfer Menu 126–127
 - Getting Files 127–128, 181
 - Sending Files 127, 181

Setup Settings 125
Starting a Session 126

G

Global Variables, Defined 169
Glossary 361–370

H

Help 41, 53
Automatic Display 38
Context-Sensitive 51
Hot Keys 296, 299–300
Access Mode 310–311
Background 249
Definition Chart 328–329

I

IF Statement 159, 160, 220–223
Inactivity Timeout
Reserved Variable 251
Setup Field 88
Index Utility 199–200
Installation, Troubleshooting 337

K

Kermit Protocol 131–137
File Transfer 131–135, 181–184
File Transfer Switches 134
Filetransfer Menu 131–132
Packet Size 93, 253, 255
Receiving Files 133–134, 182–183
Remote Commands 135–137, 215–216
Remote Menu 132, 135–137
Sending Files 133, 182
Setup Subwindow 92–95
Timeout 94, 254
Versions 131
Wildcards 133
Keyboard 50–51
ATTN Key 50–51
BHOST Settings 315, 317
BLAST Keys 50, 296, 298–299, 327–329
CANCEL Key 51
Definition Charts 327–335

Emulation Keys 329–335
Emulator Maps 296, 300
Frequently Used Keys 40–41, 50–51
Hot Keys 296, 299–300, 328–329
Mapping 296–302
Soft Keys 296, 298
User-Defined Maps 296, 301–302

Keyboard File

Creation 301–302
Reserved Variable 252
Setup Field 71

See also Keyboard

Keyboard Mapping. *See* Keyboard
Keys. *See* Keyboard

L

LAN Drivers. *See* Network Drivers

LANs 28–35

Background Mode and 17–18
Communication Port Setting 23
Network Protocols 28–35

Learn Mode 54, 160–163

Local Commands 57–58, 212–213

Local Menu 57–58

Log File

BLAST.OPT Setting 25
Error Checking 188–189
Reserved Variables 247, 257
Setup Field 71

Login

BHOST 305
Password 67, 259–260
System Scripts 193, 195
Userid 67, 270, 305

M

Menus 49–59

Access 308–311
Filetransfer 56–57, 110, 131–132
Local 57–58
Navigation through 41, 49
Offline 52–54
Online 54–56
Remote 58–59, 120–121, 135–137
Summary of 41–42

Message 56, 110, 179, 213–214

Modem
 Scripts 193, 197–198
 Setup Field 69–70
 Troubleshooting 337, 338–339
 Winmodem 339
Mouse Support 51–52, 315

N

Network Address
 BLAST.OPT Settings 25, 27
 Setup Field 67–68
Network Drivers 28–34
 ACS 29–30
 Installation Example 33–34
 IPX 31–32
 NETBIOS 32–33
 NMP 30–31
 Reserved Variable 242
 TCP/IP 33
Network Protocols 28–35
Numeric Constant, Defined 202
Numeric String, Defined 202
Numeric Value, Defined 202

O

Offline Menu 52–54
Online Demonstration and Testing Service.
See Blaster
Online Menu 54–56
OPTDIR 8–9, 17

P

Packet Acknowledgement 103–104
 Request Frequency 90, 240
 Window Size and 89, 274
Packet Size
 BHOST 100, 305
 BLAST Protocol 104, 121, 259
 Kermit Protocol 93, 253, 255
 Line Quality and 265
 Setup Field 93, 99–100
Parity
 7-Bit Operation and 104
 Blaster Setup Field 43
 Reserved Variable 259

 Setup Field 70
Password
 Reserved Variable 259–260
 Security 260
 Setup Field 67
 See also Transfer Password
Printing
 Auto Print Command 330, 331
 Autopoll Banner Files 352
 Autopoll Summary Files 352
 BHOST Settings 318
 Error Message 323
 Hot Keys 329
 Local PRINT Command 213
 LPRINT Command 226
 Print Command 58, 59, 121, 333
 Print Mode Setup Field 77
 Print Screen Command 77, 330, 331
 Printer Logging 328
 Remote 59, 119, 120, 121, 215
 Terminal Emulation 273, 290, 295, 330

Protocols

 Definition 101
 Reserved Variable 260
 Setup Field 87
 Setup Subwindows 87–99
 See also individual protocols: BLAST Protocol, FTP, Kermit Protocol, Xmodem Protocol, Ymodem Protocol, and Zmodem Protocol

R

Record, Access Menu Option 310
Registration 1–2
Remote Commands
 Autopoll 347
 BLAST Protocol 120–121, 179, 214–215
 Enabling/Disabling 91, 248
 Kermit Protocol 135–137, 215–216
Remote Control 303–319
 Access Menu 308–311
 Access Mode 306, 308–311
 Connecting to Host PC 304–306
 Disconnecting from Host PC 308
 File Transfer Only Mode 306
 Terminal Mode 306, 311–312

Remote Menu 58–59
 BLAST Protocol 110, 120–121
 Kermit Protocol 132, 135–137
 Reserved Variables 239–282
 @7BITCHN 240
 @ACKFREQ 240
 @ANSIAUTOWRAP 240
 @ANSILEVEL 240
 @APROTO 240
 @ARG_n 241
 @ATTKEY 241
 @AUTOGROUND 241
 @AUTOLFIN 241
 @AUTOLFOUT 241
 @BAUDRATE 242
 @BLASTDIR 242
 @CHARDLY 242
 @COMMPORT 242
 @COMP_LVL 243
 @CONNTIMO 243
 @CONTIMO 243
 @CTS 243
 @D/S_BITS 243
 @DATE 244
 @DATEFORMAT 244
 @DCD 245
 @DCDLOSS 245
 @DGCURSTYPE 245
 @DGDATABITS 245
 @DGPRTMODE 245
 @DGPRTWIND 246
 @EFERROR 160, 246–247
 @EFLOG 189, 247
 @EFLOGGING 247
 @ELAPTIME 247
 @EMULATE 247
 @ENABLEFS 248
 @ENABLERCMD 248
 @FILTER 248
 @FULLSCR 248
 @GROUND 248
 @HOTKEY 249
 @HPBLKTERM 249
 @HPDESTRBS 249
 @HPFLDSEP 249
 @HPINHDC2 250
 @HPINHNDSHK 250
 @HPINHWRAP 250
 @HPLINEPAGE 250
 @HPSTARTCOL 251
 @HPTERMID 251
 @INACTIVO 251
 @KBCHECK 251
 @KDELAYOS 251
 @KEYBOARD 252
 @KEYFILE 252
 @KFILETYP 252
 @KFNAMCONV 252
 @KREOPKT 253
 @KRETRY 253
 @KRPADCH 253
 @KRPADDNG 253
 @KRPKTLEN 253
 @KRSOPKT 254
 @KRTIMEOUT 254
 @KSAVEINC 254
 @KSEOPKT 254
 @KSPADCH 254
 @KSPADDNG 255
 @KSPKTLEN 255
 @KSSOPKT 255
 @KSSTIMEOUT 255
 @KWARNING 255
 @LAUNCHST 256
 @LINEDLY 256
 @LOCECHO 256
 @LOGDATEFORMAT 256
 @LOGFILE 188–189, 257
 @LOGTIMEFORMAT 257
 @LOGTIMO 257
 @MAXMEM 257
 @MODEM 258
 @NETSERVICE 258
 @NUMDISC 258
 @ONERROR 158, 258
 @ORGANS 259
 @PAKTSZ 259
 @PARITY 259
 @PASSWORD 259–260
 @PHONENO 260
 @PROMPTCH 260
 @PROTOCOL 260
 @PULSEDIAL 260
 @RBTOT 261

@RBYTES 261
 @RCOMP_LEV 261
 @RETRAN 261
 @RFAILURE 261
 @RLINEQ 261
 @RLQ 262
 @RNAME 262
 @ROPTIONS 262
 @RPACK 262
 @RPTOT 262
 @RRET 262
 @RSIZE 263
 @RSTART 263
 @RSTATUS 263
 @RSUCCESS 263
 @RTIME 263
 @RTSCTS 264
 @SBTOT 264
 @SBYTES 264
 @SCOMP_LEV 264
 @SCRFILE 264
 @SCRIPTERR 265
 @SCRLREG 175, 265
 @SETUPDIR 265
 @SFAILURE 265
 @SLINEQ 265
 @SLQ 265
 @SNAME 266
 @SOPTIONS 266
 @SPACK 266
 @SPTOT 266
 @SRET 266
 @SRTOT 266
 @SSIZE 266
 @SSTART 267
 @SSTATUS 267
 @SSUCCESS 267
 @STATUS 167–168, 204–205, 267–268
 @STATUSLN 175, 268
 @STIME 268
 @SYDESC 268
 @SYSTYPE 268
 @TIME 268
 @TIMEFORMAT 269
 @TRANSTAT 175, 269
 @TRPASSWD 269
 @TTIME 269
 @USERID 270
 @USERIF 175, 270
 @VERSION 270
 @VT8BIT 270
 @VTANSBACK 270
 @VTAUTOWRAP 271
 @VTCLRSCRN 271
 @VTCOMPRESSED 271
 @VTCURSOR 271
 @VTCURSTYPE 271
 @VTDISP132 272
 @VTHSCROLL 272
 @VTHSCROLLN 272
 @VTINTL 272
 @VTKEYPAD 272
 @VTNEWLINE 273
 @VTPRINT 273
 @VTPRINTPAGE 273
 @VTRESET 273
 @VTSTATUSLN 274
 @VTTEXTCURS 274
 @VTUSERCHAR 274
 @VTUSERKEYS 274
 @WDWSIZ 274
 @WT4ECHO 275
 @WY25LINE 275
 @WYANSBACK 275
 @WYAUTOPAGE 275
 @WYAUTOSCROLL 275
 @WYAUTOWRAP 276
 @WYBLOCKEND 276
 @WYCOMMODOE 276
 @WYCOMPRESSED 276
 @WYCURTYPE 276
 @WYDISP80 276
 @WYDSPCURSOR 277
 @WYENTER 277
 @WYEXPNDMEM 277
 @WYPAGELEN 277
 @WYPERSONALITY 277
 @WYRETURN 277
 @WYSCROLLINC 278
 @WYSEWORD 278
 @WYWRITEPROT 278
 @XLOG 278–279
 @XLTFILE 279
 @XONXOFF 279

@ZMAUTODOWN 143, 187–188, 279
 @ZMBLKLN 279
 @ZMCONVR 279
 @ZMCONVS 280
 @ZMCRC 280
 @ZMCTLESCR 280
 @ZMCTLESCS 280
 @ZMEXIST 280
 @ZMFRMLEN 281
 @ZMMANAGR 281
 @ZMMANAGS 281
 @ZMRESUME 281
 @ZMWINDOW 282
 RTS/CTS Pacing 35–36
 Reserved Variable 264
 Setup Field 86–87

S

Screen
 Command Area 38–39
 Description of 38–40
 File Transfer Status Area 39–40
 Host PC 317–318
 Scrolling Region 39, 176
 Status Line 39

Script Commands 201–238
 ASCII 205–206
 ASK 206
 CALL 168–169, 206–207
 CLEAR 207
 CLEOL 207
 CONNECT 159, 208
 CURSOR 208
 DISCONNECT 160, 208–209
 DISPLAY 157, 176, 209
 DROP 209
 ECHO 209–210
 ERRSTR 210
 FCLOSE 173–174, 210
 FILETRANSFER FILE 211
 FILETRANSFER GET/SEND 211–212
 FILETRANSFER LOCAL 212–213
 FILETRANSFER MESSAGE 213–214
 FILETRANSFER REMOTE 214–216
 FLUSH 217
 FOPENA 173, 217
 FOPENR 173–174, 217–218
 FOPENW 173–174, 218
 FREAD 173–174, 218
 FREE 219
 FREWIND 219
 FWRITE 173–174, 219
 GOTO 169, 220
 IF 159, 220–221
 IF-ELSE 222
 IF-END 160, 222
 IF-END/ELSE-END 222–223
 LCHDIR 223
 LDELETE 223
 LET 224
 LLIST 224
 LOAD 224–225
 LOCAL SYSTEM 177, 225
 LOWER 225–226
 LPRINT 226
 LRENAME 226
 LTYPE 226
 MENU 227
 NEW 227
 PUT 228
 QUIT 228
 RAISE 228
 REMOVE 229
 REPS 170, 229
 RETURN 229–230
 SAVE 230
 SELECT 230
 SET 230–231, 239
 SETTRAP 172–173, 231
 STRCAT 170–172, 232
 STRINX 171, 232
 STRLEN 171, 232
 STRTRIM 171, 233
 TCAPTURE 172–173, 233–234
 TERMINAL 234
 That Set @STATUS 204–205
 TSEND 161–162, 234–235
 TTRAP 161–162, 235
 TUPLOAD 190–192, 235–236
 UPPER 236
 WAIT 236

- WAIT CARRIER 237
- WAIT IDLE 237
- WAIT UNTIL 237–238
- WERROR 175, 238
- WRITE 175, 238
- Script File
 - Reserved Variable 264
 - Setup Field 71
- Scripting 155–192
 - Automation with 59–60, 343–359
 - Blank Lines in 166–167, 178–179, 181, 204
 - CALL Statement 168–169, 206–207
 - Capturing Text 172–173
 - Comments in 157, 204
 - Communication with Other Programs 176–177
 - CONNECT Statement 159, 208
 - Data Types 201–204
 - Downloading Text 192
 - Error Checking 162–163, 188–189
 - FILETRANSFER Statement 160, 211–216
 - IF Statement 159, 160, 220–223
 - Labels 157, 166
 - Learn Mode 160–163
 - Legal and Illegal Expressions 166–167
 - Loop in 169–170
 - Messages 179, 213–214
 - Programming Style 165–166
 - Reading Files 173–174
 - Remote Commands 179, 214–216
 - Sample 157–163
 - Screen Display 174–176
 - Syntax Rules 204
 - Text Manipulation 170–174
 - Text Transfers 190–192
 - Transfer Command Files 117–120, 179
 - Uploading Text 190–192
 - Writing Files 173–174
 - See also* Script Commands, Scripting File Transfers, and Scripts
- Scripting File Transfers 178–189
 - BLAST Protocol 178–181
 - Error Checking 188–189
 - FTP 181
 - Kermit Protocol 181–184
 - Xmodem Protocol 184–185
 - Ymodem Protocol 185–187
 - Zmodem Protocol 187–188
 - See also* Script Commands, Scripting, and Scripts
- Scripts
 - Aborting 156
 - Index Utility 199–200
 - Invoking 155–156
 - Modem 193, 197–198
 - Slave 106, 123, 198–199
 - System 193, 194–195
 - Writing 156–163
 - See also* Script Commands, Scripting, and Scripting File Transfers
- Scrolling Region 39
 - Display Control 175, 265
 - Displaying Text 176
- Security
 - @PASSWORD and 260
 - BLAST Protocol 123–124
- Session Command Window 318–319
- Setup 61–100
 - ANSI Emulation Subwindow 72–73
 - Autopoll 344–345, 346, 349
 - BHOST 305
 - BLAST Protocol Subwindow 87–92
 - Blaster 43
 - Creating 63
 - DEC VT Emulation Subwindow 74–78
 - Default 63
 - DG Emulation Subwindow 73–74
 - Directory 62–63
 - HP Emulation Setup Subwindow 82–84
 - Kermit Protocol Subwindow 92–95
 - Loading 62
 - Modifying 64
 - Protocol Subwindows 87–99
 - Removing 65
 - Subwindows 64, 72–84, 87–99
 - Terminal Emulation Subwindows 72–84
 - WYSE Emulation Subwindow 78–82
 - Zmodem Protocol Subwindow 95–99
 - See also* Setup Fields
- Setup Fields 65–100
 - 132 Compressed 28, 75, 80
 - 25th Line 78–79

7/8 Bit Controls 74
 7-Bit Channel 89
 80/132 Columns 74
 ACK Request Frequency 90
 ANSI Level 73
 Answerback 80
 Answerback Msg 76
 Attention Key 72
 Auto Page 79
 Auto Receive 98, 143
 Auto Scroll 79
 Auto Wrap 73, 76, 79
 AutoLF In 84
 AutoLF Out 85
 Baud Rate 70
 BlkTerminator 84
 Block End 81–82
 Block-Check-Type 95
 Char Delay 85–86
 Clear Screen 76
 Columns 80
 Comm Mode 81
 Connection 67–68
 Connection T/O 68
 Conversion Override 96
 CRC 98
 Cursor Keys Mode 75
 Cursor Type 73, 76, 81
 Data Bits 73
 Data/Stop Bits 71
 DCD Loss Response 89
 Delay 94
 Description 65–66
 Destructive BS 83
 Display Cursor 81
 Emulation 72, 288
 Enable /FWD and /STR 91
 Enable /OVW and Remote Cmds 91
 End-of-Packet Char 92–93
 Enter 81
 Esc All Control Chars 97
 Expanded Memory 80
 File Conversion 98
 File Management 99
 File Must Already Exist 96
 Filename Conversion 95
 Filtering 90
 FldSeparator 83
 Full Screen 84
 Horiz Scroll Inc 81
 Horizontal Scroll 75
 Inactivity Timeout 88
 Incomplete File 95
 Inh DC2(H) 82–83
 InhEolWrp(C) 83
 InhHndShk(G) 82
 Intl Char Set 77
 Jump Scroll Inc 75
 Keyboard File 71, 301, 302
 Keypad Mode 75
 Launch String 91
 Limit Block Length 97–98
 Limit Frame Length 98
 Line Delay 86
 Line/Page(D) 83
 Local Echo 84
 Local Print Option 74
 Log File 71
 Logon T/O 87–88
 Management Option 96–97
 Modem Type 69–70
 New Line 77
 Number of Disconnect Blocks 90
 Originate/Answer 68–69
 Packet Size 93, 99–100
 Pad Character 93
 Padding 93–94
 Page Length 79
 Parity 70
 Password 67
 Personality 80
 Phone Number 66
 Print Mode 77
 Print Screen 77
 Print Window 74
 Prompt Char 85
 Protocol 87
 Pulse Dialing 70
 Receive Compression Level 91
 Reset Terminal 75–76
 Resume Interrupted File 96
 Retransmit Timer 90
 Retry Limit 94
 Return 81

- RTS/CTS Pacing 86–87
- Script File 71
- Send Compression Level 91
- Size of Tx Window 98
- Start Col 83
- Start-of-Packet Char 92
- Status Line 78
- System Type 66–67
- Terminal Id 82
- Text Cursor 76
- Timeout 94
- Transfer Password 88–89
- Transfer Type 94
- Translate File 71–72, 284, 287, 290
- Use “A” Protocol 90
- User Def Keys 76
- User Pref Char Set 77
- Userid 67
- Wait for Echo 85
- Warning 95
- Window Size 89
- Write Protect 80
- Wyseword 79
- XON/XOFF Pacing 86
- See also Setup*
- Setup Subwindow. *See Setup and Setup Fields*
- Setup Window. *See Setup and Setup Fields*
- SETUPDIR
 - BLAST.OPT Setting 26
 - Environment Variable 9
- Slave Mode. *See Slave Script*
- Slave Script 106, 123, 198–199
- Sliding-Window Design 103
- Snapshot, Access Menu Option 310
- Soft Keys 296, 298
- Speaker, BLAST.OPT Setting 26
- Starting BLAST 37–38
 - Background Mode 14
 - Troubleshooting 338
- Status Line 39, 78, 175, 268
- String Constant, Defined 202–203
- String Values, Defined 203
- System Scripts 193, 194–195

T

- Technical Support 5–6
- Terminal Emulation 287–295
 - ADM3A 292–293, 333
 - ANSI 72–73, 291, 331
 - BLAST Keys 299
 - D80 292–293, 333
 - DEC VT 12, 74–78, 290–291, 330–331
 - DG 73–74, 291–292, 332
 - HP2392 82–84, 293, 334
 - IBM 3101 335
 - IBM3101 293–295
 - Keyboard Mapping 296, 300
 - Printing 295
 - Reserved Variable 247
 - Setup Field 72
 - Setup Subwindows 72–84
 - TTY 11, 289–290
 - TV920 292–293, 333
 - WYSE 78–82, 292–293, 333
- Terminal Mode 54
 - BHOST 306, 311–312
 - Local Echo 84, 256
 - Script Command 234
- Terminals. *See Terminal Emulation and Terminal Mode*
- Testing Service. *See Blaster*
- Text Transfers 145–147
 - Downloading Text 146–147, 192
 - Scripting 190–192
 - Uploading Text 145–146, 190–192
- Time Format
 - @LOGTIMEFORMAT 257
 - @TIME 268
 - @TIMEFORMAT 269
- Timeout
 - BHOST Settings 315–316
 - BLAST Protocol 107–108
 - Connection 68, 243
 - Inactivity 88, 251
 - Kermit Protocol 94, 254
 - Logon 87–88, 257
- TMP 9
- Transfer Command Files 117–120, 179
 - Autopoll 347, 350
- Transfer Password 123–124
 - Reserved Variable 269

- Setup Field 88–89
- Translate File 284–287
 - PASSTHRU.TBL 290
 - Reserved Variable 279
 - Setup Field 71–72
- Troubleshooting 337–340
- TSRs 315

U

- Uploading Text 145–146, 190–192
 - Error Detection 145
 - Flow Control 145
 - Scripting 190–192
 - Upload Menu Option 55
- User-Defined Maps 296, 301–302

V

- Variables
 - Defined 202
 - See also* Reserved Variables
- View Command 58

W

- Wildcards 112–113
 - Autopoll 347
 - BLAST Protocol 112–113
 - FTP 127
 - Kermit Protocol 133
 - Ymodem Protocol 142
 - Zmodem Protocol 143
- Winmodem 339

X

- Xmodem Protocol 140–141
 - Command Line Switches 140
 - Connection Restriction 140
 - File Transfer 140–141, 184–185
 - Filename Restrictions 143–144
 - Filetransfer Menu 56
 - Limitations 139
 - Receiving Files 141, 184–185
 - Sending Files 141, 184
- XON/XOFF Pacing 36
 - 8-Bit Transparency and 290

- ATTN Key Sequence for 328
- Reserved Variable 279
- Setup Field 86

Y

- Ymodem Protocol 141–142
 - Command Line Switches 140
 - File Transfer 141–142, 185–187
 - Filename Restrictions 143–144
 - Filetransfer Menu 56
 - Limitations 139
 - Receiving Files 142, 186
 - Sending Files 141–142, 186
 - Wildcards 142

Z

- Zmodem Protocol 142–143
 - Auto Receive 98, 143, 187–188, 279
 - File Transfer 142–144, 187–188
 - Filename Restrictions 143–144
 - Filetransfer Menu 56
 - Limitations 139
 - Receiving Files 143, 187–188
 - Sending Files 142–143, 187
 - Setup Subwindow 95–99
 - Wildcards 143

TO: BLAST Technical Support

FAX #: 919-542-0161

FROM:

Voice #:

COMPANY:

FAX #:

DATE:

IMPORTANT: Please provide us with the following information:

Your BLAST version # _____ Serial # _____

Your operating system _____ Version # _____

Where does the problem occur? (please circle)

Installation

File Transfer

Terminal Emulation

Scripting

Background

Remote Control

Other _____

Please describe the problem:

How Was It?

We would like to hear your feedback on the usefulness of this document. Your opinions can help us improve it in the future.

BLAST Professional DOS User Manual

2MNPDOS

June 2000

1. Please rate the following:

Excellent

Good

Fair

Ease of finding information

Clarity

Completeness

Accuracy

Organization

Appearance

Examples

Illustrations

Overall satisfaction

2. Please check areas that could be improved:

Introduction

Organization

Include more figures

Include more examples

Add more detail

More step-by-step procedures

Make it more concise

Make it less technical

More quick reference aids

Improve the index

3. Please elaborate on specific concerns and feel free to comment on any topics not raised previously:

Please FAX or mail these comments to us. Our contact information is listed on the title page of this manual. Thank you for your input.