Data General

Customer Documentation

Using Business BASIC on DG/UX™ and 386/ix™ Systems

Using Business BASIC on DG/UX™ and 386/ix™ Systems

093-000685-00

For the latest enhancements, cautions, documentation changes, and other information on this product, please see the Release Notice (085-series) supplied with the software.

Ordering No. 093-000685

Printed in the United States of America
Rev. 00, December 1989

Notice

Using Business BASIC on DG/UX™ and 386/ix™ Systems
093-000685-00

Revision History:	Effective with:
Original Release - December, 1989	Business BASIC for AViiON™ Systems, Rev. 1.00 Business BASIC for 386/ix™ Systems, Rev. 1.00

Preface

Data General's Business BASIC runs on the following operating systems: mapped ECLIPSE® RDOS, DG/RDOS, AOS, AOS/VS, AOS/VS II, the DG/UX™ operating system, and INTERACTIVE Systems Corporation's 386/ix™ operating system. In most instances, this guide uses the term *DG/RDOS* to refer to RDOS and DG/RDOS, and the term *AOS/VS* to refer to AOS, AOS/VS, and AOS/VS II.

UNIX® is a registered trademark of AT&T. However, since the 386/ix software product and the DG/UX software product have been derived from, or relate to, or work in conjunction with UNIX software, these two software products are sometimes referred to herein as UNIX products, solely for the purpose of improved readability. This liberty is taken in this manual only where 386/ix and UNIX, or DG/UX and UNIX, have areas of commonality or overlap, and not where the differences between them are significant.

This manual is written for Business BASIC applications programmers who will work in a UNIX environment and for the person who must perform Business BASIC system-management functions in that environment. For a list of subjects covered in the manual, see the next section.

Organization of This Manual

This manual includes seven chapters and one appendix.

Chapter 1 discusses the facets of the UNIX file system with which you should be familiar if you plan to write or port applications that will run in a UNIX environment.

Chapter 2 explains what the Business BASIC resource limits are for a programmer working on a UNIX system.

Chapter 3 deals with calling auxiliary Business BASIC programs and the UNIX shell from your main program.

Chapter 4 explains how to generate a tailored Business BASIC interpreter.

Chapter 5 discusses how to start and stop the Business BASIC interpreter. This discussion includes information on bringing up an obituary-handling program, a resource lock server, and a page-table daemon; on managing queues and devices; and on setting environment variables.

Chapter 6 deals with system security.

Chapter 7 explains how to port existing Business BASIC applications to a UNIX environment.

Appendix A lists Business BASIC, DG/RDOS, AOS/VS, and UNIX error messages.

Document Set

Using Business BASIC on DG/UX and 386/ix Systems is part of a four-manual set that describes the language, its utilities and subroutines, and how the system is set up. The other manuals in this set are

- *Learning Business BASIC* (093-000684)
- *Commands, Statements, and Functions in Business BASIC* (093-000351)
- *Subroutines, Utilities, and the Business BASIC CLI* (093-000389)

Reader, Please Note

Data General manuals use certain symbols and styles of type to help clarify the information they contain. The Data General symbol and typeface conventions used in this manual are defined below. You should familiarize yourself with these conventions before reading the manual.

This manual also presumes the following meanings for the terms "command line," "format line," and "syntax line." A command line is an example of a command string that you should type verbatim; it is preceded by a system prompt. A format line shows how to structure a command; it shows the variables that must be supplied and the available options. A syntax line is a fragment of program code that shows how to use a particular routine; some syntax lines contain variables.

Convention	Meaning
boldface	In command lines and format lines: Indicates text (including punctuation) that you type verbatim from your keyboard. The names of all commands, statements, functions, subroutines, utilities, files, and directories are also printed in boldface.
`constant width font`	Represents a system response on your screen. Syntax lines also use this font.
italic	In format lines: Represents variables for which you supply values, for example, the names of your directories and files, your username and password, and possible arguments to commands.
[ ]	In format lines: These brackets surround an optional argument. Don't type the brackets; they only set off what is optional.
...	In format lines and syntax lines: Means you can repeat the preceding argument as many times as desired.

Contacting Data General

- If you have comments on this manual, please use the prepaid Comment Form that appears at the back. We want to know what you like and dislike about this manual.
- If you require additional manuals, please use the enclosed TIPS order form (USA only) or contact your local Data General sales representative.

Telephone Assistance

If you are unable to solve a problem using any manual you received with your software, and you are within the United States or Canada, contact the Data General Service Center by calling 1-800-DG-HELPS for toll-free telephone support. The center will put you in touch with a member of Data General's telephone assistance staff who can answer your questions.

Free telephone assistance is available with your warranty and with most Data General service options. Lines are open from 8:30 A.M. to 8:30 P.M., Eastern Time, Monday through Friday.

For telephone assistance outside the United States or Canada, ask your Data General sales representative for the appropriate phone number.

End of Preface

Contents

Chapter 1—The UNIX File System

Chapter 2—Operating System Dependent Limits

Chapter 3—Communication Between Programs

Chapter 4—Generating a Business BASIC Interpreter

Chapter 5—Starting and Stopping the BASIC Interpreter

Chapter 6—System Security

Chapter 7—Porting Applications to a UNIX System

Appendix A—Error Messages

Index

Tables

Table

Figures

Figure

Chapter 1
The UNIX File System

This first chapter discusses the facets of the UNIX file system with which you should be familiar if you plan to write or port applications that will run in a UNIX environment. The chapter covers the following subjects:

- Filenames
- The Structure of the File System
- File Access Privileges
- System Files
- Accessing Files from a Business BASIC Program
- Opening a File Exclusively
- Deleting Files

Filenames

Although a Business BASIC program generally identifies a file by its channel number (which is mapped to a UNIX file descriptor), the underlying operating system identifies a file by its filename. This name can be made up of any character except null (\0) and slash (/). However, many other characters have a special meaning to the UNIX shell, which is the counterpart of the DG/RDOS and AOS/VS CLIs. These include

@ # ^ & () ‘ [] \ | ; ’ :

Other characters are not printable. Also, Business BASIC historically has limited the characters allowed for use in filenames. Therefore, Business BASIC requires that you use only the characters listed below when forming filenames:

- A to Z (uppercase letters)
- a to z (lowercase letters)
- 0 to 9 (digits)
- _ (underscore)
- . (period)

Business BASIC also allows you to use the question mark (?) and the dollar sign ($) as filename characters; however, we suggest that you do not use them because they have special meanings for the shell. The shell interprets the question mark as a special character or template that matches any one character in a filename, and a leading dollar sign indicates to the shell that you want it to substitute the value of a variable for the name of a variable.

In addition to the limits on legal filename characters, UNIX also restricts the length of filenames. The 386/ix system accepts filenames of up to only 14 characters, while a DG/UX system allows 255 characters. If you name a file while in Business BASIC, you must restrict the name to one less than the maximums stated above. Business BASIC uses a shadow file to indicate that a given file is a link file, and the name of this shadow file is the same as that of the link file except that it begins with a period.

One further caveat. Because UNIX is case sensitive—that is, the names **file1** and **FILE1** refer to different files—all references to a file in your program and the name by which the operating system knows the file must be exact matches. The one exception to this rule occurs when you invoke Business BASIC with the **-P** option, which causes the interpreter to convert AOS/VS-style pathnames to their UNIX counterparts, and lowercase letters in simple filenames to uppercase letters. For more information on pathname conversion, see the section "Accessing Files from a Business BASIC Program" later in this chapter.

The Structure of the File System

The UNIX file system includes ordinary files, special files, and directory files. Ordinary files contain text, data, or executable programs. Special files are device drivers that allow you to communicate with peripheral devices such as terminals, printers, and disk and tape drives. Directory files contain, or store lists of, ordinary files, special files, and other directories (called subdirectories).

These files are structured as a tree. At the root of the tree is a directory named / (this directory is also called the *root* directory). This directory contains ordinary files and directories that are internal nodes themselves. For example, in the directory structure shown in Figure 1-1, the root directory contains the program file **unix** and the subdirectories **bin, dev, etc,** and **udd**. The first of the three subdirectories are terminal nodes—that is, they only contain pointers to ordinary and special files—but **udd** contains the directory files **grp1** and **grp2**. The directory **grp1**, as you can see, contains further subdirectories. There is no limit on the depth of the tree structure, nor is there any limit on the number of ordinary and special files a directory may contain.

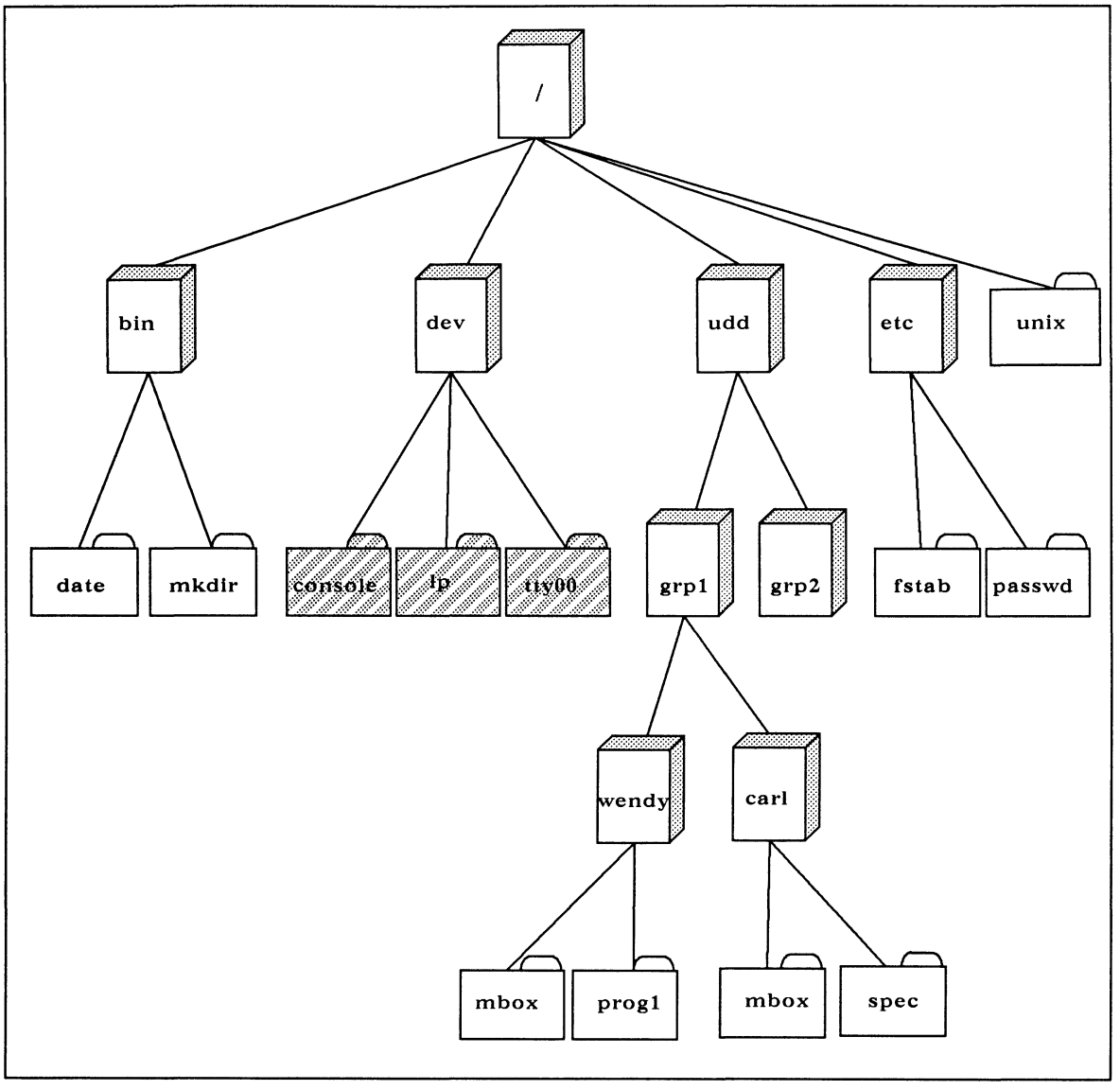
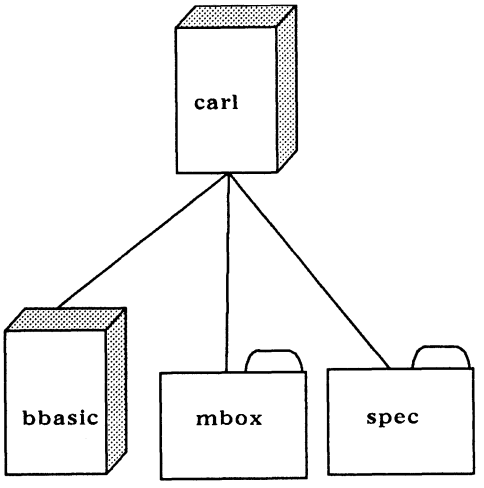


Figure 1-1 The UNIX File System

When you log in to the system, you are normally executing the UNIX shell, your interface to the operating system. By using it, you can create files and directories, move files about in the file system, or start a program such as the Business BASIC interpreter. The directory you are in is known as your home directory. For Carl, this would be the directory **carl**. At the moment, this directory is also your current directory.

At present, Carl has only two ordinary files in his home directory. However, as stated above, there is no limit on the depth of the directory tree structure, so Carl can create a directory subordinate to **carl**. He does this by using the format line **mkdir** *directory-name*. For example, if he enters the command line **mkdir bbasic**, the system creates the directory **bbasic** so that the subtree whose root is **carl** now looks like this:



Carl can enter the new directory by using the command line **cd bbasic**.

Pathnames

To refer to a file, you must use a *pathname* or rely on one of the search mechanisms discussed in the next section.

A pathname is a list of two or more filenames that guides you through the file system to the file you are interested in. All of the filenames except the last must be the names of directory files. The final filename can be the name of an ordinary file, a special file, or a directory: this is the file you want to refer to. The pathname can begin either with the root directory or with a directory immediately below or above your current directory.

If the pathname begins with the root directory, it is called a *full* or *absolute pathname*. For example, if the file system shown in Figure 1-1 were real, the full pathname for the file **prog1** would be **/udd/grp1/wendy/prog1**. In this pathname, the first slash represents the root directory, and the subsequent slashes are separators. There is no separator between the symbol for the root directory and the name of the next file in the path.

If the pathname does not begin with the root directory, it is called a *relative pathname* because you do not specify the starting point for the path. That starting point is always your current directory.

Most often, a relative pathname is similar to an absolute pathname in that the path it describes moves down through the file system. Refer again to Figure1-1. If Wendy were working in the directory **grp1** and wanted to execute **prog1**, she would tell the system to execute **wendy/prog1**.

Relative paths, however, can also move up through the file system, or up and then down. For instance, if Wendy were working in the directory **wendy** and wanted to list the files in the directory **udd**, she would ask the system to list the contents of

```
../..
```

The symbol .. (two periods) indicates to the system that you want to move up one level in the directory structure. Since Wendy has used the symbol twice, she would move up two levels. To display the contents of **spec**, Wendy would need to provide the system with a path that points up to **grp1** and then down to **spec**: **../carl/spec**.

Search Rules

If you are not running the Business BASIC interpreter, the search rules on your system are determined by the contents of the UNIX environment variable PATH. This variable contains a list of directories that the shell will search each time you attempt to execute a program. Note that the shell will not find non-executable files. The shell searches the first directory stored in **PATH** first, then the second, and so on.

A typical uncustomized PATH variable may contain the string

```
:/bin:/usr/bin
```

This string indicates that the shell should first search the current directory, then **/bin**, then **/usr/bin**. The current directory is denoted by a colon that is not preceded by a directory name. The second colon serves as a separator.

You can make a permanent addition or alteration to the contents of PATH by editing the file **.profile** or **.login** in your home directory. You can also change the value of PATH by using a shell command or executing a script that contains such a command; however, the latter type of change lasts only until your shell process is terminated.

If you are running the interpreter, the search rules for your system are determined by the contents of the environment variable BBPATH. The content of BBPATH resembles that of PATH, and the two variables play similar roles; however, there are three important differences between the way the UNIX shell uses PATH and the way the interpreter uses BBPATH. First, while the shell searches the directories in PATH for executable files only, Business BASIC searches the directories in BBPATH for all types of files. Second, if the UNIX shell sees a relative pathname, it assumes that the first (or only) directory in the pathname is immediately subordinate or superior to the current directory; it does not consult PATH. On the other hand, if Business BASIC sees a relative pathname and the first directory in the pathname does not exist immediately below or above the current directory, the interpreter checks to see whether the initial directory in the pathname is subordinate or superior to a directory in BBPATH. Third, while the shell does not search the current directory unless PATH includes that directory, Business BASIC always searches the current directory.

For an explanation of how to assign a value to BBPATH, see Chapter 5.

Symbolic Links

Symbolic links are not a feature of the standard System V UNIX® operating system. However, the environment in which you run Business BASIC programs must provide for symbolic links because Business BASIC's logical file system makes extensive use of these link files. Symbolic link files are also desirable because they help save storage space when more than one person wants to access the same file. For example, let's assume that Wendy has stored the program **prog1** in the directory **/udd/grp1/wendy** and that three other people need to run the program. If the three others create symbolic links in the directory from which they execute the interpreter or in a directory whose name is stored in the environment variable BBPATH, only one copy of the program needs to be stored, as opposed to four. Symbolic links also reduce the number of directories you need to store in BBPATH.

For these reasons, Business BASIC uses a shadow file to indicate that a file is a link file. The name of this shadow file has the form *.link-file-name*. As an example, Carl can create a link to **prog1** by entering the following commands:

```
* ER=0
* STMC 21,ER,"prog_wendy","/udd/grp1/wendy/prog1"
```

After he issues these commands, three files will exist:

- **prog_wendy** (the symbolic link file)
- **.prog_wendy** (the shadow file)
- **prog1** (the resolution file, in Wendy's directory)

In this example the link file points to the name of a program, but it is also legal for the link file to point to a second symbolic link. The maximum link depth is 16.

CAUTION: To delete a link file, use the Business BASIC CLI command **UNLINK** or **STMC 35**. If you use the command **DELETE** *link-file-name*, you will delete the resolution file and be left with a symbolic link that points to a nonexistent file. At this point, if you try to access the deleted resolution file through the link, you receive an error message stating that the file does not exist. One further note. Since Business BASIC handles links differently from the operating system, you should create and delete Business BASIC links only from Business BASIC.

File Access Privileges

The UNIX file-protection scheme is based on the following model. Users are classified as one of three types: the owner of the file, other members of the owner's group, and users outside the group. In addition, any type of user may have any combination of three types of file access: read (r), write (w), and execute (x). See Figure 1-2.

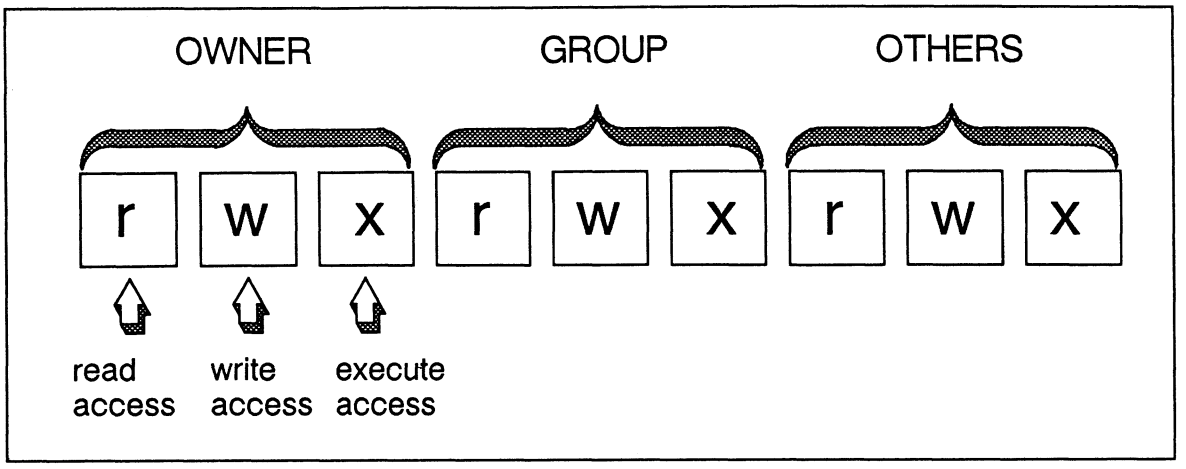


Figure 1-2 The UNIX File-Protection Scheme

To determine the types of access that the three types of users have to a particular file, move to the directory that contains the file using the **cd** command and then enter the command **ls -l**. The program **ls** will list on your monitor one line for each file in the directory, and each line will look similar to this one:

```
-rwxr-xr-x   1 carl       bbasic      59743  Oct 12 10:58  spec
```

For the moment ignore the first character in the line and look at the next nine characters. The first three indicate that Carl has permission to read and write data to **spec**, and if the file were an executable program, he would also have permission to execute it. The next three characters mean that the other members of the group have read and execute access to the file. The last three characters indicate that users outside the group also have read and execute permissions.

If **spec** were a directory instead of a text file, you would interpret the codes *r*, *w*, and *x* differently. *R* would indicate that a user had permission to use the **ls** command to list the contents of the directory. *W* would mean that the user had permission to create new files in the directory and to delete existing ones. And *X* would indicate that a user had permission to move to the directory (using the **cd** command).

There are two ways to change the access permissions associated with a file, but the easiest is to use the shell's **chmod** command and provide as arguments a three-digit octal number and the name of the file whose access permissions you want to change. The **chmod** command takes the numeric argument you supply and looks at it as a binary number occupying nine bits, which correspond to the nine characters we looked at above. For instance, if the high-order bit is set to 1, the program grants the owner of the file read access to the file named on the command line; if that bit is 0, the owner does not receive read access. The program looks at the remaining 8 bits in the same way. Thus, the command to change the access bits for the file **spec** so that users outside the group have no access to the file is

chmod 750 spec

Refer to Figure 1-3 below.

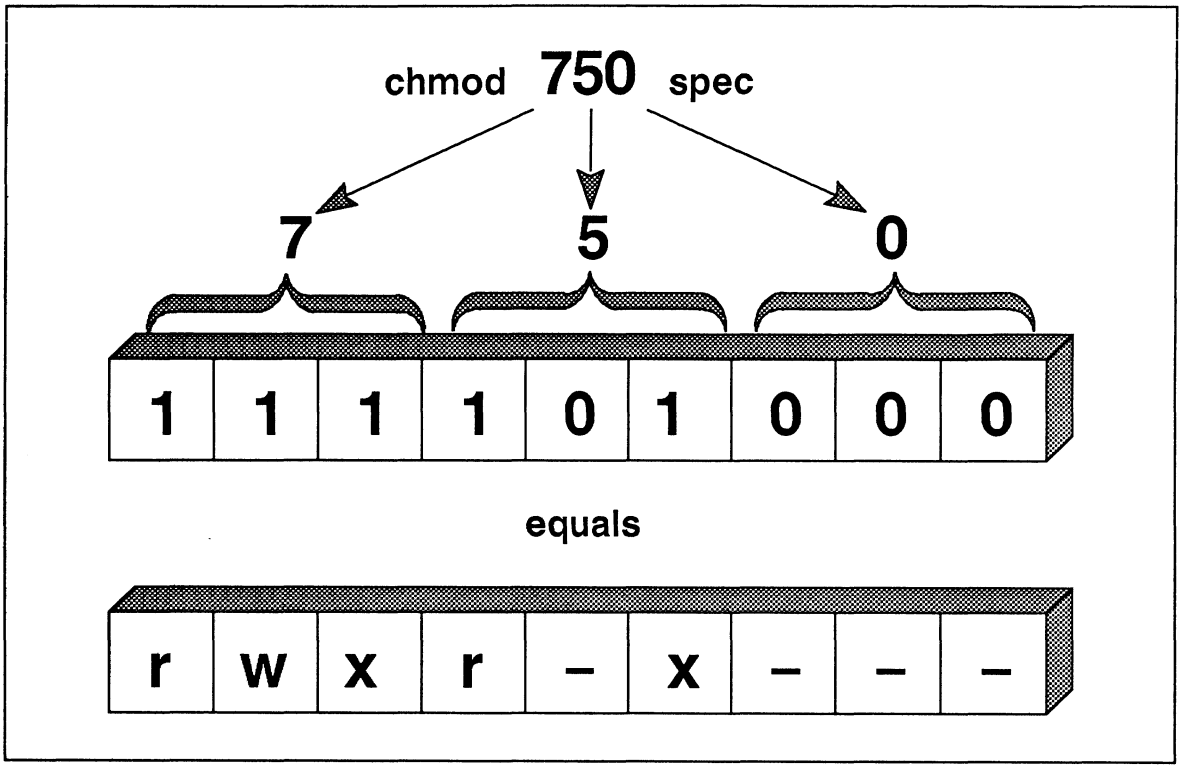


Figure 1-3 Changing a File's Access Bits

For further information on the **chmod** command, see your user's reference manual.

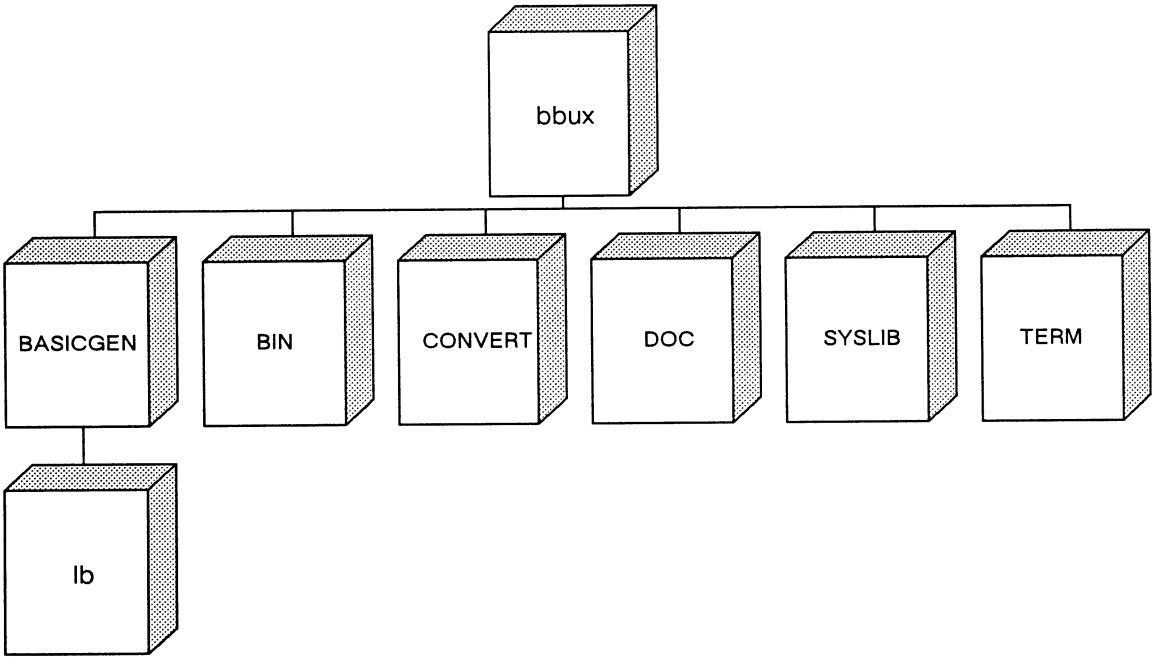
It is important to be able to check and alter a file's access privileges because certain Business BASIC I/O statements require specific privileges. The following list details these requirements:

- **OPEN FILE.** Requires write access to the directory containing the file you want to open or the directory in which you want to create the file. If you are opening an existing file, you must also have read or read and write access to the file, depending on the mode of access you select.
- **DELETE.** Requires write access to the parent directory of the file you want to delete, but not to the file itself.
- **READ FILE, INPUT FILE, LOAD, ENTER.** Require read access to the target file.
- **WRITE FILE, PRINT FILE, SAVE, REPLACE.** Require write access to the target file.
- **KADD, KDEL.** Require write access to the index file.

For further information on these statements, see the manual *Commands, Statements, and Functions in Business BASIC.*

System Files

Once you have installed Business BASIC, the core of the directory structure that contains the Business BASIC system files will look like this:



All these directories contain system files. The best way to determine the function of any given file is to examine its filename extension. Table 1–1 below shows the filename extensions that you may come across in these directories and explains their meanings.

Table 1-1 Filename Extensions

Extension	Meaning
BASICGEN	
.libraries	Contains a list of all the libraries supplied with Business BASIC.
.revision	Contains the revision level of your Business BASIC software.
no ext.	The Business BASIC script **bbasic_build**.
lb	
.a	A Business BASIC library, a collection of C object files.
.c	A C source file.
.h	A C header file.
.o	A C object file.
BIN	
.ex	An example of a UNIX script file that you may want to create.
no ext.	The Business BASIC script **bbux_mgr** or an executable file. The executables in this directory include the default runtime system and the following Business BASIC daemons: **obit**, **ptd**, and **rlsx**.
CONVERT	
.BA	A Business BASIC listing file for a utility that you may need to use to move your programs from a DG/RDOS or AOS/VS system to a UNIX system.
.CLI	An AOS/VS CLI macro that you may use if you are moving programs from an AOS/VS to a UNIX system.
DOC	
.BA	A Business BASIC listing file for a utility.
.fl	A list of all the files included in your Business BASIC release.
.rn	The Business BASIC Release Notice.

(continued)

Table 1-1 Filename Extensions

Extension	Meaning
SYSLIB	
.DB	A database file in a logical file database structure.
.DS	A file containing a description of each field in a PARAM file record.
.ER	An error-message file.
.PS	A list of Business BASIC reserved words.
.RM	A version of the FM utility that uses commands instead of function keys.
.SL	A Business BASIC subroutine.
.T6	A part of the file maintenance utilities.
.TB	A table file used with the FM utility.
.VL	A volume label file (part of the logical file database structure).
no ext.	A Business BASIC utility.
TERM	
no ext.	A modified TERMINFO packet.

(concluded)

Accessing Files from a Business BASIC Program

UNIX is case sensitive. Therefore, if you attempt to open a file with the statement

```
OPEN FILE (1,6) "WEEKLYRPT"
```

and the name of the file is actually **weeklyrpt**, the open will fail. Conversely, if you try to open a file with the statement

```
OPEN FILE (1,6) "weeklyrpt"
```

and the name of the file is **WEEKLYRPT**, the open will fail unless you used the **-P** option when you executed the runtime system. The **-P** option causes Business BASIC to translate lowercase filenames to uppercase. For more information on the **-P** option, see Chapter 5.

In general, it is your responsibility to ensure that the name your program uses to access a file and the name by which the file is known to the UNIX file system are an exact match. However, ensuring such a match can be difficult in a case where the **-P** option is of no help and you are reading the name of a file from a terminal or a disk file into a string variable. In this instance, you must use **STMA 14** to make sure that the name of the file stored in the string variable is converted to lowercase or uppercase, as appropriate. For example, consider the following segment of code:

```
110 INPUT "Enter a filename: ",A$
120 STMA 14,A$,3
130 OPEN FILE(1,6),A$
```

Let's assume that the names of your data files consist of lowercase letters. Even if the user of your program enters uppercase letters in response to the prompt `Enter a filename`, line 120 converts those characters to their lowercase counterparts before the name is handed to the **OPEN FILE** statement in line 130. You can convert lowercase characters to uppercase by using the format line **STMA 14,***string-variable*,**0**.

If you have moved programs from a DG/RDOS or AOS/VS system to your UNIX system, you may have a couple of other problems accessing files in the UNIX file system. For example, if a program contains a DG/RDOS or AOS/VS pathname, the program will not execute on your UNIX system because the separator used in pathnames on DG/RDOS and AOS/VS systems is a colon, while on UNIX systems, it is a slash. Depending on how you have capitalized the names of your directories and files on your UNIX system, you may also have problems matching the capitalization of the names of directories and files in your program with the capitalization of the names of those files as they are known to the UNIX operating system.

If you wish to avoid making changes to your sources to account for such problems, follow these guidelines. First, create for your programs and data files a directory structure that mimics the directory structure that existed on your DG/RDOS or AOS/VS system. Also, give the directories in this structure and your programs and data files uppercase names. Thus, if the pathname to a file on your AOS/VS system were **:UDD:SALES:DEPT37**, the pathname to the same file on your UNIX system should be **/UDD/SALES/DEPT37**. If you set things up this way, you can use the **-P** option mentioned above when you start your interpreter to request that Business BASIC convert all AOS/VS-style pathnames to their UNIX counterparts. For example, if the interpreter sees the statement

OPEN FILE(0,0),":udd:sales:dept37"

it will convert the pathname to

/UDD/SALES/DEPT37

Note that the interpreter replaces colons with slashes and converts all lowercase characters to uppercase.

If you are moving programs over from a DG/RDOS or AOS/VS system and don't want to make changes to your sources, you may also have problems with **OPEN FILE** statements that open devices or queues. To overcome this difficulty, use the **-m** *device-map-name* option when you start Business BASIC. If you have set up your device map file correctly, using this

option will cause the interpreter to replace the names of your DG/RDOS and AOS/VS devices and queues with their UNIX counterparts. See Chapter 5 for more information on managing devices and queues.

Opening a File Exclusively

Business BASIC allows you to open a file exclusively with the **OPEN FILE** command. However, the ability to open a file in this mode is not a feature of the standard UNIX System V operating system. Therefore, for users to open files exclusively, they must adhere to a protocol established by Business BASIC. Because this protocol depends on the read and write locking of bytes of files, it might not be supported by the Network File System (NFS) on a remote machine. This implementation might conceivably affect exclusive or nonexclusive opens in a couple of other ways as well.

First, closing a UNIX file descriptor destroys all lock information for the file with which the descriptor is associated, even if the process that closed the descriptor still has the file opened with another descriptor. Therefore, since Business BASIC uses read and write locks of certain bytes of the file to determine the type of open, it cannot close the file descriptor and free it for further use until you have tried to close all the descriptors for a file. For example, if you reconfigure the UNIX kernel according to the instructions in your Business BASIC Release Notice, 106 file descriptors are available to your Business BASIC process. The interpreter uses from four to six of these descriptors as soon as you start it, leaving you with from 100 to 102. Let's say that your process has 100 descriptors to use. If you open the same file for nonexclusive use on all 100 of these descriptors, close 99 of them, and then try to open another file, you will receive an error because there will be no descriptors available to your process.

Second, it is possible to exhaust all the file descriptors allotted to your process by attempting to open a file that you already have opened. Suppose that instead of opening the same file with all 100 of your descriptors, you open the file for exclusive use (one descriptor) and then try to open the file for nonexclusive use 99 times. Each of these attempted opens will cause an error because the file is already opened exclusively. Any further attempt to open the file will fail because no more file descriptors are available to your process.

These restrictions are necessary for Business BASIC to be able to provide exclusive opens on UNIX platforms and should not pose a problem under normal circumstances.

Deleting Files

If the filename argument you supply on your **DELETE** (delete a file) command line is the name of a link file, Business BASIC deletes the resolution file and leaves the link file unchanged. You can delete the link file using the BASIC CLI command **UNLINK** or the statement **STMC 35**.

One other thing to note. If one process opens a file and another deletes the file, the process that deletes the file does not receive an error.

End of Chapter

Chapter 2
Operating System Dependent Limits

When you are developing or porting Business BASIC applications, it is important to be aware of certain restrictions that Business BASIC or UNIX place on you: the maximum number of lines your program may include, for example, or the maximum length of a variable name. This chapter is an attempt to present the most important of these limitations in one place. The chapter comprises the following sections:

- Program Lines and Size
- Variables and Arrays
- The File System
- The Common Area
- Nested Statements

Program Lines and Size

A Business BASIC program written to run in the UNIX environment can contain line numbers ranging from 1 to 99,999. The maximum length of a single line is 256 characters, including the terminating New Line.

The largest program that can run on a UNIX system, including program and data space, is 512 Kbytes. Program statements and your data can share this space in any way that suits your purpose. That is, you can have a large program and a small amount of data, or the converse.

Variables and Arrays

A program can contain a maximum of 8,192 variables. The name of each variable must begin with a letter and can contain up to 32 characters chosen from the following list: uppercase or lowercase letters, digits, and the underscore. In addition, a variable name may end with one of the following data-type specifiers: %, #, &, or $. This specifier is not counted in the length of the name. By default the UNIX Business BASIC interpreter allots 8 bytes for each numeric variable (quadruple precision). The maximum dimension for a string variable is a little less than 512 Kbytes.

Arrays may contain either numeric or string data and may have from one to eight dimensions.

The File System

This section contains information on UNIX filenames and open-file limits.

A filename is the name by which a file is known to the UNIX operating system. UNIX allows a filename to be composed of any character except null (\0) and slash (/); however, since many of these characters are not printable or have special meaning to the UNIX shell, and since Business BASIC historically has limited the characters available for filename use, UNIX Business BASIC requires that you use only the characters listed below when forming filenames:

- A through Z (uppercase letters)
- a through z (lowercase letters)
- 0 through 9 (digits)
- _ (underscore)
- . (period)

Business BASIC also allows you to use the question mark (?) and the dollar sign ($) as filenames characters; however, we suggest that you do not use them because these characters have special meaning for the shell. UNIX interprets the question mark as a special character or template that matches any one character in a filename, and a leading dollar sign tells UNIX that you want it to substitute the value of a variable for the name of a variable.

Besides these limits on legal filename characters, UNIX places restrictions on the length of filenames. The 386/ix system accepts filenames of up to only 14 characters, while DG/UX allows 255 characters. If you are naming a file from Business BASIC, you must restrict the name to one less than the maximums stated above. Business BASIC uses a shadow file to indicate that a given file is a link file, and the name of this shadow file is the same as that of the link file except that it begins with a period.

The number of files one process can have open is normally 60. However, if you reconfigure the kernel of the operating system according to the instructions in your Business BASIC Release Notice, you will increase this limit to 106. When you start Business BASIC, the interpreter uses four descriptors to open the files **stdin, stdout, stderr,** and **BASIC.ER**. It may also use two more descriptors, one for the push file and one for the system channel. This means that a program can open a maximum of 100 to 102 files.

The Common Area

The common area, which you use to pass data between programs, is 2 Kbytes in size.

Nested Statements

Business BASIC allows nesting of the following statements: **DEF**, **FOR/NEXT**, **DO/WHILE/UNTIL** and **GOSUB/RETURN**.

- The **DEF** statement allows you to define your own functions. Once you have defined one function, you can use it in your definition of a second, and so forth. The maximum number of user-defined functions that can appear in one statement is 26.
- In a program in which **FOR/NEXT** loops are nested, the maximum number of loops that can be involved is 32.
- You may also have up to 32 levels of nested **DO/WHILE/UNTIL** loops.
- In a program in which a **GOSUB** statement transfers control to a subroutine, which in turn transfers control to a second subroutine, and so on, the maximum number of **GOSUB** levels allowed is 32.

End of Chapter

Chapter 3
Communication Between Programs

A Business BASIC program can call, and pass information to, other Business BASIC programs or the UNIX shell. The following sections discuss these calls in detail.

Calling a Business BASIC Program

To transfer control to a Business BASIC program that you have saved on disk, include in the calling program either the **CHAIN** or the **SWAP** command. If you use a statement of the form **CHAIN** "*filename*, that statement causes the called program to execute and the calling program to be discarded. Control never returns to the calling program. A **SWAP** statement also passes control to a second Business BASIC program; however, it does not cause the calling program to be discarded. The calling program is stored in memory or on disk, and when the called program stops, the original program resumes execution at the line following the **SWAP** statement.

You can pass data between the calling and the called programs in several ways.

- If the data you need to pass between Business BASIC programs is one word or less, you can pass the data using **STMA 1** and **STMA 2**.
- If you need to pass 2 Kbytes or less of information between programs, you can pass the data through your common area. In the calling program you use a **BLOCK WRITE** statement to write data to the common area, and in the called program, you use a **BLOCK READ** statement to read that data.
- If you need to pass more than 2 Kbytes of information, a good strategy is to write the data you want to make available to a second program to a temporary file. The called program can then read the data from the temporary file.

In the example below, the initial program, **MAKE_ARRAY3** creates a three-dimensional numeric array and fills it with the integers 1 through 1000. The program also writes the number of dimensions in the array, the size of each dimension, and the content of each element in the array to a temporary file named **PASS_DATA**. The program then swaps to a second Business BASIC program, **INVERT**, whose purpose is to invert the elements in a one- or multi-dimensional array.

This second program reads from **PASS_DATA** the information it needs to calculate the number of elements in the array that was created and filled by the original program. **INVERT** then reads the values of the elements from the three-dimensional array into a one-dimensional array, placing the first value read in the last element of its array, and so on. Once the values are in reverse order, **INVERT** writes them to a temporary file, again called **PASS_DATA**.

Finally, control returns to **MAKE_ARRAY3**, which reads the values 1000 through 1 into its three-dimensional array.

Here is the code for **MAKE_ARRAY3**:

```
00010 LET DIMS=3
00020 LET DIM1=9
00030 LET DIM2=9
00040 LET DIM3=9
00050 LET ELEMENT=1
00060 DIM ARRAY[DIM1,DIM2,DIM3]
00070 OPEN FILE[10,0],"PASS_DATA"
00080 WRITE FILE[10],DIMS,DIM1,DIM2,DIM3
00090 FOR SUB1=0 TO DIM1
00100   FOR SUB2=0 TO DIM2
00110     FOR SUB3=0 TO DIM3
00120       LET ARRAY[SUB1,SUB2,SUB3]=ELEMENT
00130       WRITE FILE[10],ARRAY[SUB1,SUB2,SUB3]
00140       LET ELEMENT=ELEMENT+1
00150     NEXT SUB3
00160   NEXT SUB2
00170 NEXT SUB1
00180 PRINT "ELEMENT #1 =";ARRAY[0,0,0]
00190 PRINT "ELEMENT #1000 =";ARRAY[9,9,9]
00200 SWAP "INVERT"
00210 POSITION FILE[10,0]
00220 FOR SUB1=0 TO DIM1
00230   FOR SUB2=0 TO DIM2
00240     FOR SUB3=0 TO DIM3
00250       READ FILE[10],ARRAY[SUB1,SUB2,SUB3]
00260     NEXT SUB3
00270   NEXT SUB2
00280 NEXT SUB1
00290 CLOSE FILE[10]
00300 PRINT "ELEMENT #1 =";ARRAY[0,0,0]
00310 PRINT "ELEMENT #1000 =";ARRAY[9,9,9]
00320 DELETE "PASS_DATA"
00330 END
```

And this is the code for **INVERT**:

```
00010 LET DIM1,DIM2,DIM3,DIM4,DIM5,DIM6,DIM7,DIM8=0
00020 POSITION FILE[10,0]
00030 READ FILE[10],DIMS
00040 IF DIMS=1 THEN READ FILE[10],DIM1
00050 IF DIMS=2 THEN READ FILE[10],DIM1,DIM2
00060 IF DIMS=3 THEN READ FILE[10],DIM1,DIM2,DIM3
00070 IF DIMS=4 THEN READ FILE[10],DIM1,DIM2,DIM3,DIM4
00080 IF DIMS=5 THEN READ FILE[10],DIM1,DIM2,DIM3,DIM4,DIM5
00090 IF DIMS=6 THEN READ FILE[10],DIM1,DIM2,DIM3,DIM4,DIM5,DIM6
00100 IF DIMS=7 THEN READ FILE[10],DIM1,DIM2,DIM3,DIM4,DIM5,DIM6,DIM7
00110 IF DIMS=8 THEN READ FILE[10],DIM1,DIM2,DIM3,DIM4,DIM5,DIM6,DIM7,
DIM8
00120 LET ELEMENTS=(DIM1+1)*(DIM2+1)*(DIM3+1)*(DIM4+1)*(DIM5+1)*
(DIM6+1)*(DIM7+1)*(DIM8+1)
00130 DIM ARRAY[ELEMENTS]
00140 FOR SUB1=ELEMENTS-1 TO 0 STEP -1
00150   READ FILE[10],ARRAY[SUB1]
00160 NEXT SUB1
00170 CLOSE FILE[10]
00180 OPEN FILE[10,0],"PASS_DATA"
00190 FOR SUB1=0 TO ELEMENTS-1
00200   WRITE FILE[10],ARRAY[SUB1]
00210 NEXT SUB1
00220 END
```

Running **MAKE_ARRAY3** produces the following results:

```
* RUN
* ELEMENT #1 = 1
* ELEMENT #1000 = 1000
* ELEMENT #1 = 1000
* ELEMENT #1000 = 1

*
```

Calling the UNIX Shell

In addition to being able to communicate with another Business BASIC program, your program can call and pass information to the UNIX shell if you invoke the interpreter with the **-c** option. You can also select which type of shell you wish to communicate with, the Bourne shell or the C shell, by setting the environment variable BBSHELL. For more information on this subject, see Chapter 5.

There are limitations on the amount of data that you can pass between programs: your program can pass just one command line to the UNIX shell, and the shell cannot return any data. Still, this type of call can be very useful. For instance, in the previous section, we looked at a program called **MAKE_ARRAY3** that uses a temporary file named **PASS_DATA**.

In line 330 the program uses a **DELETE** statement to remove this file. Suppose, however, that the program had created 20 temporary files. It would be inefficient to use 20 **DELETE** statements to remove these files. It would be much cleaner to ensure that each temporary file has the filename extension **.TMP** and then to pass the shell a command to delete all 20 files.

To do something like this, you use the **SHELL** statement, whose format is shown below:

SHELL [*error-code, command*]

Error-code must be a numeric variable and must be initialized. If the call to the shell is successful, the value of *error-code* after the call will be –1. If the call is unsuccessful because you did not start the interpreter with the –c option, the error code returned will be 0. If the call fails for some other reason, *error-code* will contain a number associated with a UNIX error message. *Command* can be a string variable or a string literal; in either case the data passed to the shell should be a single shell command line terminated by a null byte.

The code to delete the 20 temporary files mentioned above might look like this:

```
00330 LET ERRCODE=0
00340 SHELL ERRCODE,"rm *.TMP<0>"
00350 IF ERRCODE=-1 THEN PRINT "Temporary files deleted."
00360 IF ERRCODE<>-1 THEN
00370   PRINT "Error in calling shell."
00380   PRINT "Error code = ";ERRCODE
00390 END IF
00400 END
```

After the shell has deleted the temporary files, the execution of your program resumes at line 350.

The command line you pass to the shell can contain more than one command if you place a semicolon after each command except the last. For instance, to display the name of the current directory and a list of the files in that directory, you could use these Business BASIC commands:

```
* LET ERRCODE=0
* SHELL ERRCODE,"pwd;ls -l<0>"
```

You can also use the **SHELL** command to call the shell without passing a command to it. In this instance, you execute a shell process and can enter any number of commands. To return to your program, you log out of the shell in the usual way, by typing **exit, logout**, or **Ctrl-D**.

For more information on the **SHELL** statement, see the manual *Commands, Statements, and Functions in Business BASIC*.

End of Chapter

Chapter 4
Generating a Business BASIC Interpreter

After you finish installing Business BASIC according to the instructions in your Release Notice, a Business BASIC interpreter named **bbasic** exists in the directory **/usr/opt/bbux/BIN**. (It is possible on a 386/ix system that Business BASIC may have been installed in a directory other than **/usr/opt/bbux**.) This interpreter is a development system, uses the new implementation of **SWAP**, and gives all users the ability to use privileged statements. In addition, the person who installed Business BASIC on your system may have generated a custom interpreter while installing the software. If an interpreter that meets your needs exists, there is no reason for you to build a new interpreter, so you can skip the remainder of this chapter. If the current interpreter does not fit your needs, follow the instructions below to build a tailored interpreter.

NOTE: If you are working on a 386/ix system, make sure that the Software Development System package has been installed on your system before you attempt to generate a new interpreter.

Starting the System-Generation Script

The script you use to build a tailored interpreter is called **bbasic_build** and is located in the directory **/usr/opt/bbux/BASICGEN**. You can start the script by following one of the two procedures mapped out below.

To call the script from the shell, follow these directions. Log in as **root** and move to the directory **/usr/opt/bbux/BASICGEN** using the **cd** command. Also, make sure that one of the directories stored in the environment variable PATH is the current directory. To make this determination, enter the **env** command. Your terminal will display a list similar to the following one:

```
# env
HOME=/
HZ=100
LOGNAME=root
PATH=/bin:/etc:/usr/bin
TERM=vt100
TZ=EST5EDT
```

In this instance, the current directory is not one of the directories listed in PATH. If it were present, it would be indicated by a colon with no directory name before it, by a period, or by two consecutive colons. To add the current directory to the existing list of directories, use this command if you are working in the Bourne shell:

```
# PATH=:$PATH
```

If you are working in the C shell, use these commands:

```
# set path=:$PATH
# setenv PATH $path
```

Now start the system-generation script by entering the following format line:

bbasic_build [*interpreter-name*]

NOTE: If the current directory is not listed in PATH, you do not have to change the contents of that environment variable to start the script. Instead you can use the command **./bbasic_build**. The period before the slash represents the current directory.

After you enter this command, the system prompts you to answer a series of questions to obtain the information it needs to build your interpreter. These questions are discussed in the section "Building a Tailored Interpreter" below.

You can also execute the **bbasic_build** script by calling the **sysadm** utility. If you are working on a 386/ix system, issue this command:

```
sysadm packagemgmt
```

The utility will print an analysis of how your disk space is being used and then display this menu:

```
                    PACKAGE MANAGEMENT

   1  bbasic_mgmt     Business BASIC management menu
   2  lpmgmt          add line printer
   3  tcpipmgmt       extended networking utilities menu

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP, ^ to GO BACK, q to QUIT:
```

Select option 1, and you will see the Business BASIC management menu:

```
                BUSINESS BASIC MANAGEMENT

   1  config_mgmt     Generate a Business BASIC interpreter.
   2  daemon_mgmt     Business BASIC Daemon management menu.
   3  run_bb          Run the Business BASIC interpreter.
```

Selecting option 1 at this point will start the **bbasic_build** script, which poses the questions discussed in the next section.

If you are working on a DG/UX system, issue the command

sysadm bbasic_mgmt

This command will take you directly to the BUSINESS BASIC MANAGEMENT menu, where you can select option 1 to execute the **bbasic_build** script.

Building a Tailored Interpreter

This section lists the questions that **bbasic_build** asks and makes recommendations about how you should answer those questions.

When you execute the **bbasic_build** script, you see the following message and prompt:

```
Business BASIC generation utility revision x.xx.xx.

Continue with Business BASIC generation? [y] (y,n):
```

The default to the script's first question is shown in square brackets, and a list of possible answers is shown in parentheses. If you want to generate a new interpreter, you should type y or just press the New Line key.

If you started the system-generation script from the shell and provided an interpreter name on your command line, you will not see the next prompt. However, if you started the utility without providing such an argument, the script asks you to name the new interpreter:

```
Enter system name [bbasic]:
```

If you press the New Line key, your interpreter will be named **bbasic**. To select another name, enter a valid filename and then press the New Line key. Legal filenames are discussed in Chapter 2 in the section "The File System."

The next question concerns a log file:

```
Do you want to specify a log name? [y] (y,n):
```

It's advisable to request a log file because, if you do, the command lines used to compile your parameter file and to link your interpreter are recorded in this file. If you later report a problem with your interpreter, the Data General engineer working with you on your problem will find these commands useful while trying to reproduce your problem. To request a log file, type y or just press the New Line key.

If you request a log file, **bbasic_build** also needs to know the name of that file. The default name is the name of your interpreter plus the extension **.log**. So if the name of your system is **bbasic**, the prompt you see will look like this:

```
Enter log file name [bbasic.log]:
```

To opt for the default log file name, press the New Line key, and to select a different name, enter a valid filename followed by a New Line.

At this point, the system-generation script asks the first of the three questions that actually determine how your system will be built:

```
Do you want a development system? [y] (y,n):
```

If you answer **n**, users of this system will only be able to run existing programs saved in metacode format. They will not be able to use the **ENTER** command to bring a listing file on disk into working storage or to list a program. If you need these capabilities or plan to do any program development on your system, answer **y** or just press the New Line key.

The script's second important question concerns the **SWAP** command:

```
Do you want to use the new SWAP implementation? [y] (y,n):
```

In the past a **SWAP** statement caused the interpreter to write the current program to disk before running the program being swapped to. Beginning with Revision 1.00 of Business BASIC for UNIX systems, however, you have the option of specifying that the program containing the **SWAP** statement be saved in memory. The number of levels you can push depends solely on the amount of memory available to your Business BASIC process. The advantage of using the new implementation of **SWAP** is that your programs that contain **SWAP** statements will execute faster. The disadvantage is that the new implementation requires more memory than the old.

The third important question involves the use of privileged statements:

```
Do you want all users to be able to use privileged statements? [n]
(y,n):
```

The privileged statements being referred to are the **STMB**s, the **STMC**s, the **STME**s, and the **STMU**s. If you respond **n**, you will achieve a level of security by preventing nonprivileged users from working at the system-call level (see Chapter 6); however, nonprivileged users can load and run programs that contain the statements referred to above. If all the programmers on your system need access to these statements, the simplest thing to do is to answer **y** to this question.

At this juncture, you should see messages similar to following ones

```
 Compiling the parameter file.
e_sw_genp_gd.c: 53: SW_PRIVILEGED_USER_DEFAULT redefined


 Now linking xxxxxx. Please wait.
Would you like to move /usr/opt/bbux/BASICGEN/x to /usr/opt/bbux/BIN/x
? [y] (y,n):
```

The interpreter supplied with Business BASIC resides in **/usr/opt/bbux/BIN**, so if you want to keep all of your interpreters in the same directory, answer **y** or just type a New Line. If you answer **y**, the interpreter will be moved immediately unless an interpreter of the same name already exists in the directory **BIN**. In this case, you are given the option of overwriting the existing interpreter. You will then see this message:

```
Business BASIC generation complete.
```

At this point, if you started the **bbasic_build** script from the shell, you are returned to the shell prompt. If you used the **sysadm** utility to get to the script, you will see this message:

```
Press the RETURN key to see the bbasic_mgmt menu [?,^,q]:
```

Type **q** to return to the shell prompt. You are now ready to execute the interpreter you have built. For instructions on how to do so, see the next chapter.

End of Chapter

Chapter 5
Starting and Stopping the BASIC Interpreter

This chapter explains how to perform the following tasks:

- Start the obituary-handling program, **obit**, and the resource lock server, **rlsx**
- Manage devices and queues
- Set up a series of environment variables
- Execute the interpreter
- Terminate a Business BASIC program, the runtime system, **rlsx**, and **obit**
- Handle an abnormal termination of the interpreter

NOTE: Like Chapter 4, this chapter assumes that Business BASIC has been installed in the directory **/usr/opt/bbux/BIN**. On DG/UX systems, the software must be loaded in this directory; however, on 386/ix systems, it is possible that the software may reside in a different directory.

Managing obit and rlsx

This section explains what the **obit** and **rlsx** programs do, how to start them, and how to monitor the status of the programs.

The obit Program

The purpose of the program **obit** is to clean up all interprocess-communication structures left behind by Business BASIC processes, **rlsx** processes, or **ptd**, the page-table daemon, after those processes have terminated. In particular, **obit** cleans up useless message queues, semaphores, and shared memory segments. The program also removes the file **.lock_table**, which is created by **rlsx**.

The rlsx Program

Record locking for Business BASIC processes is coordinated through the use of a resource lock server, **rlsx**. This program keeps lock information in a database that both Business BASIC processes and **rlsx** can access. This arrangement allows a Business BASIC process to enter a lock in the shared database if the record it needs is available. If the lock request is successful, no interprocess communication is necessary. If the request is unsuccessful, your process must ask **rlsx** to handle the locking.

Business BASIC processes try to minimize the interprocess-communication traffic between **rlsx** and other Business BASIC processes. For example, when a Business BASIC process unlocks a record that another process is waiting for, the unlocking process sends a message directly to the waiting process.

There may be more than one **rlsx** running on your system. One **rlsx** may be accessed by test programs under development, for example, while another is accessed by live applications. If there is more than one **rlsx** on your system, each user must make sure that his environment variable RLSX_DIR is set so that his programs will access the correct **rlsx**. Setting this environment is discussed later in this chapter in the section "Setting Environment Variables."

No one should terminate a Business BASIC process that is modifying the **rlsx** database or updating an index file. Doing so could corrupt the index file. Thus, users should avoid the careless use of the quit or kill signal to terminate Business BASIC when record locking is being employed.

Starting obit and rlsx

When the person who installed Business BASIC on your system installed the software, he may have requested that the **obit** and **rlsx** processes be started whenever you boot your operating system. In this case, the processes should be running now. To determine whether they are running, enter the **ps**, process status, command:

```
$ ps -e

. . . COMMAND
      sched
      init
      .
      .
      .
      obit
      rlsx
      obit
```

If **obit** and **rlsx** are listed in the rightmost column of the display, the processes are running. You should read the rest of this section, however, since in the future you may want to restart one of the programs without rebooting your operating system. If the programs are not running, you should start them using a script called **bbux_mgr**. You can execute this script in either of two ways.

To execute the script from the shell, you must be logged in as **root**, and you will probably want to move to the directory **/usr/opt/bbux/BIN**, where the script is located. Before entering the name of the script, use the **env** command to see whether the environment variable PATH contains the current directory. If it does, you can start the script by entering the command

```
# bbux_mgr
```

If the current directory is not listed in PATH, use this command:

```
# ./bbux_mgr
```

The period before the slash tells the UNIX operating system to look for the script in the current directory.

After a moment your screen will clear, and you will see the script's main menu.

```
Main Menu

(1) Bring obit up
(2) Bring obit down
(3) Check status of obit

(4) Bring rlsx up
(5) Bring rlsx down
(6) Check status of rlsx

(7) Quit

Enter option number:
```

You can also run the **bbux_mgr** script by calling the **sysadm** utility. If you are working on a 386/ix system, issue this command:

```
sysadm packagemgmt
```

The utility will print an analysis of how your disk space is being used and then display this menu:

```
                        PACKAGE MANAGEMENT

    1  bbasic_mgmt     Business BASIC management menu
    2  lpmgmt          add line printer
    3  tcpipmgmt       extended networking utilities menu

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP, ^ to GO BACK, q to QUIT:
```

Select option 1, and you will see the Business BASIC management menu:

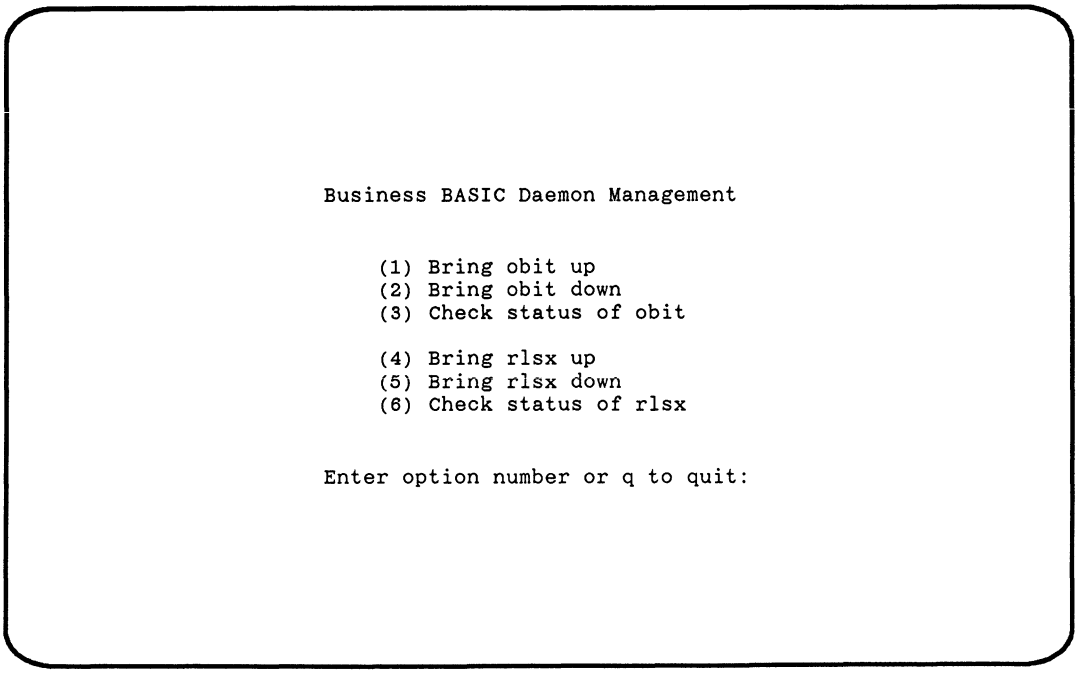
```
                    BUSINESS BASIC MANAGEMENT

    1  config_mgmt     Generate a Business BASIC interpreter.
    2  daemon_mgmt     Business BASIC Daemon management menu.
    3  run_bb          Run the Business BASIC interpreter.
```

Selecting option 2 at this point will start the **bbux_mgr** script, which displays a menu called Business BASIC Daemon Management.

If you are working on a DG/UX system, issue the command

```
# sysadm bbasic_mgmt
```

This command will take you directly to the Business BASIC Daemon Management menu shown below:



Regardless of whether you started **bbux_mgr** directly from the shell or through the **sysadm** utility, you should now follow the directions below.

It is important to bring up **obit** first, so select option 1. You will see the following prompt:

```
Enter obit directory:
```

We recommend that you choose the directory **/usr/opt/bbux/BIN**. You may either enter this pathname followed by a New Line or, since you are working in this directory, enter a period (which represents the current directory) followed by a New Line.

You will then be prompted for the maximum number of Business BASIC users. The default answer is 10. If you will have fewer or more than 10 users, enter the appropriate number.

Once you have indicated a maximum number of users, type a New Line in response to the prompt `Hit <return> to continue`. The Main Menu will reappear.

After starting **obit**, you can execute **rlsx** by selecting option 4. The script will prompt you for the name of the directory in which you brought up **obit**:

```
Enter directory obit was brought up in: [/usr/opt/bbux/BIN]
```

or

```
Enter directory obit was brought up in: [.]
```

If you started **obit** in the current directory, as was suggested above, press the New Line key to accept the default answer. You will then be asked for an **rlsx** directory.

```
Enter rlsx directory:
```

If the default answer `[/usr/opt/bbux/BIN]` or `[.]` is displayed after the prompt, just press the New Line key; otherwise, enter the pathname **/usr/opt/bbux/BIN** or a period and press the New Line key.

Finally, the script asks you to specify a size in bytes for the shared memory segment that will serve as the lock table. The default answer is 2048. You can tune this parameter by entering a number other than 2048. This number need not be a multiple of 512. If you enter a number less than 2048, you will conserve system resources. If you enter a number greater than 2048, you will be able to lock more records, and the system will have to collect garbage less often.

At this point, both **obit** and **rlsx** are running, and you are ready to leave the script. To do so, press the New Line key to reach the script's main menu. Then select option 7 if you started **bbux_mgr** from the shell, or type **q** and a New Line if you started the script using the **sysadm** utility. Taking the appropriate action will return you to the shell prompt.

Checking the Status of obit and rlsx

Once you have started **obit** and **rlsx**, you can use the **bbux_mgr** script to check on the status of those programs. When checking the status of **obit**, the script checks to see whether the **obit** process is running and whether it has access to the following interprocess-communication structures: a shared memory segment, a semaphore, and a message queue. When checking the status of **rlsx**, the script determines whether **rlsx** has access to the structures mentioned above and whether the file **.lock_table** exists.

To start the script, use the directions prescribed in the previous section. Then select option 3 to check the status of **obit** or option 6 to check on **rlsx**. After you make your selection, the script will display one line of information on your screen. If the line begins with the word OK, the program you are interested in is running normally. On the other hand, a message such as the following one indicates that something is wrong:

```
rlsx: bad rlsx status; sem(s=510), no shm, msg(q=414)
```

If you see such a message, perform the following steps:

1. Choose option 5 from **bbux_mgr**'s main menu to stop **rlsx**. (If you must stop both **rlsx** and **obit**, stop **rlsx** first.)
2. Check the status of **rlsx** again. If its status has not changed, use the shell command **ipcrm** to remove the semaphore numbered 510 and the message queue 414 from the table that lists allocated interprocess-communication resources. The exact command line to use is

   ipcrm -s 510 -q 414

   To remove a shared memory segment, use the option **-m** followed by the appropriate number.
3. Restart **rlsx**. (If you must restart both **rlsx** and **obit**, restart **obit** first.)

Managing Devices and Queues

Directing output from a Business BASIC program to a device is a straightforward procedure. The following code does exactly what you would expect:

```
00010 OPEN FILE[1,6],"/dev/ttyp0"
00020 WRITE FILE[1],"Hello, world"
```

It writes the words "Hello, world" on the the screen of the terminal known to UNIX as **ttyp0**.

The only problem you can run into when you execute this code is that Business BASIC does not have permission to write to this device. If this problem occurs, you or your system manager should move to the directory **/dev** and then use the **chmod** command to give the interpreter write access to the device. Since the **/dev** directory is re-created each time UNIX

is booted, you may want to automate the process of changing a device's permission bits. To do so, create in the directory **/etc/rc2.d** a script that contains your **chmod** command. The name of this script file should consist of an *S* followed by a two-digit number. The script with the lowest number runs first. We recommend that you use a number in the high 90's.

You may also want to add the directory **/dev** to the contents of the environment variable **BBPATH**. If you take this step, you can shorten line 10 of the program shown above to read

```
00010 OPEN FILE[1,6],"ttyp0"
```

As far as managing queues goes, there are two main things you need to be aware of. First, you can specify the default queue by using the **-q** option on the command line you use to start Business BASIC. Second, you can define the default output queue as returned by **STMA 9,3** by editing a device map file and starting Business BASIC with the **-m** option. (A sample device map file named **DEVICE_MAP** exists in the **SYSLIB** directory.) Simply make the first non-comment line in the file follow this format:

prt_queue *queue-name* *queue-name*

For example:

prt_queue lp lp

NOTE: You can change the default queue by using **STMA 10,1**.

If you plan to run programs developed in DG/RDOS or AOS/VS Business BASIC, the management of this device map file becomes even more important. The map file allows programs to use such statements as `OPEN FILE[1,2],"@LPT"` or `OPEN FILE[0,7],"@MTB0:0` successfully on UNIX.

Each line of the file is either a comment line, identified by a number sign (#) in the first column, or a line containing three fields separated by spaces or tabs. The first field contains one of three keywords: **prt_queue**, which identifies a queue entry; **prt_device**, which identifies a printer device; or **tape_device**, which identifies a tape device. The second field is the filename used in an **OPEN FILE** statement, and the third field is either the pathname of a device or the name of a queue established by you or your system manager.

Here is a sample line from a device map file:

prt_queue @LPT lpt

This line indicates that Business BASIC should construe any reference to **@LPT** as a reference to the queue **lpt**. Therefore, if the interpreter executes a statement such as `OPEN FILE[1,2],"@LPT"`, any subsequent statements that write data to channel 1 will send output to the **lpt** queue.

Here is another example:

```
prt_device      $LPT          /dev/tty11
```

If Business BASIC executes the statement **OPEN FILE[2,2],"$LPT"**, the interpreter maps the reference to **$LPT** to **/dev/tty11**. Subsequent writes to channel 2 will send output directly to the device **tty11**.

More than one filename used in an **OPEN FILE** statement may resolve to the same device or queue. For example, a device map file could contain the following lines.

```
prt_device      @LPT          /dev/tty13
prt_device      $LPT          /dev/tty13
prt_device      fastprint     /dev/tty13
```

In this case, these statements would all refer to the same device:

```
OPEN FILE[1,1],"$LPT
OPEN FILE[2,1],"fastprint
OPEN FILE[3,1],"@LPT
```

Setting Environment Variables

Before you invoke a Business BASIC interpreter, you must set a series of Business BASIC and UNIX environment variables. In the Bourne shell, you set an environment variable using commands similar to these:

```
PATH=:/usr/bin:/usr/opt/bbux/BIN:/bin
export PATH
```

In the C shell, you use a command similar to this one:

```
setenv PATH :/usr/bin:/usr/opt/bbux/BIN:/bin
```

The Business BASIC environment variables are listed and discussed in Table 5-1 below. The relevant UNIX environment variables are listed in Table 5-2.

Table 5-1 Business BASIC Environment Variables

Environment Variable	How to Set
BBDEFACL	Set to a three-digit octal number that defines the access privileges you want to assign to files you create while in the interpreter. For instance, if you want to give read, write, and execute access to yourself, members of your group, and others, set the variable to 777. (For further information on this subject, see the section "File Access Privileges" in Chapter 1 or the entry for **chmod** in your user's reference manual.) If you do not set this variable, the default value Business BASIC uses is 700. For BBDEFACL to work correctly, you must also set the UNIX system parameter **umask** to 0. To do this, add the line **umask 0** to the file **.profile** or **.login** in your home directory. Or you can set the value of **umask** for all users by adding the line **umask 0** to the file **/etc/profile** or **/etc/login.csh** (DG/UX systems only).
BBPATH	Enter a list of paths to directories that you want Business BASIC to search for user program and data files. BBPATH must also list the path to the directory **BIN**, which contains the page-table daemon, and the path to the directory **SYSLIB**, which contains the error message file **BASIC.ER** and the Business BASIC utilities. Business BASIC will not run if it cannot find the page-table daemon and the error message file, so you must set BBPATH before bringing up the interpreter. You need not include the current directory in BBPATH because Business BASIC searches the current directory automatically.
BBSHELL	Set to **/bin/sh** or **/bin/csh** if you want to specify whether the interpreter should execute the Bourne shell or the C shell when you issue the **SHELL** command. If you do not set this variable, the **SHELL** command will execute your initial login program.
BBTERMTYPE	Set to 6 if you do not want to use Business BASIC's SCREENEDIT feature. If this variable is set to any other value or if the variable does not exist, SCREENEDIT is turned on.
OBIT_DIR	Set to the path of the directory in which you brought up **obit**. You must set this variable before you can start the interpreter.
RLSX_DIR	Set to the path of the directory in which you brought up **rlsx** if you want to use the record locking capabilities of Business BASIC. If you set RLSX_DIR, **rlsx** must be running in the directory you specify before you can execute Business BASIC.

Table 5-2 UNIX Environment Variables

Environment Variable	How to Set
PATH	Add to the list of directories stored in this variable the path to the directory that holds your Business BASIC interpreter, usually **/usr/opt/bbux/BIN**. The argument $PATH represents the current contents of PATH.
TERM	Enter the name of the TERMINFO packet that corresponds to your terminal type. If you are working at a DASHER/386™ system console, specify **at386_bbux**. This packet gives you support for function keys and shifted function keys. If you are working on a D216 or D216E terminal, enter **vt100_bbux**. If you are working on a D412 or D462 terminal, enter **vt220_bbux**. This packet also provides support for function keys and shifted function keys. After you have set both the TERMINFO and TERM environment variables, you must issue the command **tput init**.
TERMINFO	Enter the pathname **/usr/opt/bbux/TERM**. This pathname guides the system to one of the modified TERMINFO packets mentioned above. After you have set both the TERMINFO and TERM environment variables, you must issue the command **tput init**.

The example below shows the exact commands you might enter to set your environment variables if you are working in the Bourne shell:

```
$ BBDEFACL=744
$ export BBDEFACL
$ umask 0
$ BBPATH=/usr/opt/bbux/SYSLIB:/usr/opt/bbux/BIN:/usr/opt/bbux
$ export BBPATH
$ BBSHELL=/bin/sh
$ export BBSHELL
$ OBIT_DIR=/usr/opt/bbux/BIN
$ export OBIT_DIR
$ RLSX_DIR=/usr/opt/bbux/BIN
$ export RLSX_DIR
$ PATH=:/usr/bin:/usr/opt/bbux/BIN:/bin
$ export PATH
$ TERMINFO=/usr/opt/bbux/TERM
$ export TERMINFO
$ TERM=vt100_bbux
$ export TERM
$ tput init
```

If you are working in the C shell, you might use commands similar to these:

```
% setenv BBDEFACL 744
% umask 0
% setenv BBPATH /usr/opt/bbux/SYSLIB:/usr/opt/bbux/BIN:/usr/opt/bbux
% setenv BBSHELL /bin/csh
% setenv OBIT_DIR /usr/opt/bbux/BIN
% setenv RLSX_DIR /usr/opt/bbux/BIN
% setenv PATH :/usr/bin:/usr/opt/bbux/BIN:/bin
% setenv TERMINFO /usr/opt/bbux/TERM
% setenv TERM vt100_bbux
% tput init
```

Note that after assigning values to an environment variable, or a series of variables, in the Bourne shell, you should issue a command that follows this format:

export *environment-variable ...*

The **export** command ensures that the environment variables you set will be set not only in the current shell process, but in all subordinate shells as well. In the C shell, the **setenv** command serves a similar purpose.

Since it would be inconvenient to set environment variables each time you log in, we recommend that you store your assignment commands in your **.profile** or **.login** file. You can do this either by editing the file **profile.ex**, which is part of the Business BASIC release and resides in the **/BIN** directory, or by adding them to the **.profile** or **.login** file supplied with your UNIX system. One of these files resides in your home directory.

If you choose the former option, follow the directions in the file **profile.ex**. If you choose the latter option, use the example below as a guide.

The following example shows how you could use the **vi** editor to set the Business BASIC environment variables in the file **.profile** in your home directory. First, issue a command similar to the following one to make sure that you are in your home directory:

```
cd /usr/ann
```

Then begin your **vi** editing session by entering this command:

```
vi .profile
```

At this point, your screen will clear. Then **vi** will display as much of the contents of **.profile** as will fit in the first window. If all of the file fits on the first screen, you will see one or more lines at the bottom of the screen that are blank except for a tilde in the first column. If the screen is full of text, type **Ctrl-F** to see the second window of text. Repeat this key sequence until you see the last line of **.profile**. Your screen should look something like this:

```
#
#
# PATH=$PATH:"your_path"
#
HZ=100
export MAIL PATH TERM HZ
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

Now type the command **j**, which moves your cursor down one line, until the cursor is sitting on the first character of the last line of the file, that is, directly above the first tilde. Then type the command **o** to open the succeeding line for text entry. You can now enter the commands that set your environment variables and export their names and values just as if you were working at a typewriter. After you have typed the last character you want to enter, press the Esc key. Doing so puts you back in command mode. Then enter the save-and-quit command, **:wq**, and a New Line.

In the future, when you log on, the commands in **.profile** will be executed automatically. So by the time you run your initial program, usually the UNIX shell, your environment variables will be set appropriately.

Invoking the Interpreter

Once you have assigned values to the appropriate environment variables, you are ready to formulate a command line that will start Business BASIC. This command line must include the name of the interpreter you want to invoke (if the path to the directory that contains your runtime system is stored in the UNIX environment variable PATH) or a full pathname that points to the interpreter. The command line may also contain one or more of the options listed in Table 5-3. These options are case sensitive, so you must enter them exactly as shown.

Table 5-3 Options for the Business BASIC Command Line

Option	What It Does
-C	Allows Business BASIC to recognize a subset of the control characters that programmers have embedded in strings and sent to terminals running in DG mode. If you use this switch, the interpreter translates each supported character so that it has the desired effect on terminals not running in DG mode.
-E double	Causes your quadruple-precision system to emulate a double-precision system.
-E triple	Causes your quadruple-precision system to emulate a triple-precision system.
-H	Instructs the interpreter to run the program **HELLO** when you start Business BASIC.
-P	Tells the interpreter to convert AOS/VS-style pathnames to UNIX pathnames. If you use this switch, Business BASIC converts colons in pathnames to slashes and any lowercase letters in pathnames and filenames to their uppercase counterparts. You must also use this option, together with the **-c** option, if you plan to use the Business BASIC CLI.
-S	Causes the interpreter to display a status line at the top of the screen. The information on the status line includes the name of the program in working storage, the date, the time, and your working directory.
-W	Suppresses parser diagnostic messages, those that attempt to pinpoint the exact nature or location of a syntax error. You will still receive a general error message when an error occurs.
-c	Allows you to use the **SHELL** statement and, when used with the **-P** option, the Business BASIC CLI, a program that performs many operating system functions.
-d	Enables you to terminate the runtime system from within your Business BASIC process by entering **Ctrl-** or your terminal's assigned quit character. For further information on this subject, see the section "Terminating the Runtime System" later in this chapter.
-m *dev-map*	Tells Business BASIC where to find the device map file that it should consult if it does not recognize the filename argument to an **OPEN FILE** statement. For more information on the device map file, consult the section "Managing Queues and Devices."

(continued)

Table 5-3 Options for the Business BASIC Command Line

Option	What It Does
-p *blocks*	Defines the maximum size (in blocks) of the programs you can create. This switch is useful if you are doing cross-development work because Business BASIC systems running on the DG/RDOS and AOS/VS operating systems do not provide nearly as much program space as do interpreters on UNIX systems. You can also use this option to allow more users on a system where a shortage of memory is limiting the number of users who can run the interpreter.
-q *queue*	Sets the default queue.
-s *program*	Starts the Business BASIC program named as soon as you log on. The argument *program* must be a program that has been saved or replaced.
-t	Causes Business BASIC to log you out of the system after 10 minutes of inactivity.
-u	Disables the bell that Business BASIC sounds if you go 10 minutes without entering any data. This option also prevents the interpreter from clearing the input buffer if you go 10 minutes without pressing the New Line or Return key.
-w	If you used the -s option, this option logs you out of Business BASIC after your initial program has run.

(concluded)

A few sample command lines follow:

1. **bbasic -c -P -s myprog**

   This command causes the interpreter supplied on your release media to execute the program **myprog** and then continue running. The **-c** and **-P** options give you access to the Business BASIC Command Line Interpreter. The **-c** option also gives you the ability to use the **SHELL** statement.

2. **bbasic -H -E double**

   If you issue this command, the interpreter executes the **HELLO** program as it comes up. While running, it emulates a double-precision system.

3. **bbasic -u**

   This command starts the interpreter and ensures that you will have unlimited time to enter a command or program statement.

If you plan to use the same command line each time you start the Business BASIC interpreter, you may want to include that command line in your **.profile** or **.login** file, just as you did the assignment commands for your environment variables. Or you can put the command line in a separate script.

Stopping Programs

This section provides instructions for terminating a Business BASIC program, the runtime system, and the daemons **rlsx** and **obit.**

Terminating a Business BASIC Program

To terminate a Business BASIC program without terminating the runtime system, enter one of the two Business BASIC interrupt characters.

The primary interrupt key is the Esc key. This key, however, does not produce a real-time interrupt; the character it produces has no special meaning to the UNIX operating system. It produces an interrupt only when your Business BASIC program has a read posted.

To provide a real-time interrupt character, Business BASIC, during initialization, sets its secondary interrupt key to be the same as the key UNIX recognizes as the interrupt key for your terminal. To find out which key UNIX regards as your interrupt key, enter the shell command **stty -a** (386/ix systems) or **stty all** (DG/UX systems). Then scan the output on your display for the phrase `intr =`. The character following the equal sign is your terminal's interrupt character. Most often, this character is Ctrl-C. You can change your terminal's interrupt character by entering a command of the form **stty intr** "*character*".

NOTE: Do not set your terminal's interrupt character to be the Esc key. Doing so may cause the shell or Business BASIC to produce unpredictable results.

Terminating the Runtime System

To stop the current program and the runtime system and return to the UNIX shell (or to log out if the interpreter was your initial program), you can use any of the following commands:

- If you used the **-d** option when you executed Business BASIC, enter **Ctrl-** or your terminal's assigned quit character. To determine what your terminal's present quit character is, enter the command **stty -a** (386/ix systems) or **stty all** (DG/UX systems). This command displays the currently selected options for your terminal. Scan this output for the phrase `quit =`. The character following the equal sign is your terminal's quit character. (If the character is preceded by a caret, your quit character is a control character; for example, the sequence ^| represents Ctrl-|.) You can change your terminal's assigned quit character by entering a command of the form **stty quit** "*character*".

  Once you enter this quit character, the runtime system terminates without cleaning up and may leave ISAM files in an undefined state. The message `User abort` is written to the file **stderr** (standard error), and then control returns to the shell. The shell is notified via an exit status code of the signal that terminated the interpreter.

- Issue the command **kill -3** from another terminal's shell process. This action has the same effect as entering **Ctrl-**.

- Issue a **kill -15** command from another terminal's shell process. This command has the same effect as **kill -3** except that the parent of your interpreter, your terminal's shell, is notified that the interpreter was terminated by the signal SIGABRT. This signal causes your shell to try to write a core dump.

- Issue a **kill -9** command from another terminal's shell process. This command kills both the current program and the runtime system immediately, and control returns to your terminal's shell. The exit code sent to your shell process is that for SIGKILL. Use this command only as a last resort because it prevents any cleanup of either ISAM files or terminal I/O modes and may leave both in unusable states.

Terminating rlsx and obit

To terminate **rlsx** and **obit**, you run the same script that you used to start these processes, the **bbux_mgr** script, which is located in the directory **/usr/opt/bbux/BIN**. Make sure that you are logged in as **root**, move to **/usr/opt/bbux/BIN**, and enter the command

bbux_mgr

If the shell cannot find the script, add the current directory to the contents of the environment variable PATH and then call the script again. As **bbux_mgr** begins executing, you see the following menu:

```
Main Menu

(1) Bring obit up
(2) Bring obit down
(3) Check status of obit

(4) Bring rlsx up
(5) Bring rlsx down
(6) Check status of rlsx

(7) Quit

Enter option number:
```

Stop **rlsx** first by selecting option 5 and then entering the pathname of the directory in which **rlsx** is running:

```
Enter option number: 5
Enter rlsx directory: /usr/opt/bbux/BIN
OK


Hit <return> to continue
```

Press the New Line key to return to the Main Menu. Then select option 2 and enter the pathname of the directory in which **obit** is running. After terminating **obit**, select option 7 to return to the shell.

Runtime System Failure

Business BASIC can terminate abnormally for a number of reasons: because of a power failure, because the UNIX operating system terminates abnormally, or because of an interpreter problem. If the power to your system was interrupted or the UNIX operating system crashed, you should, of course, restore power to your computer if necessary and reboot the UNIX system. Then follow the steps enumerated below to restart Business BASIC.

1. Check to see whether two **obit** processes and one **rlsx** process are running. To do this, enter the command

   ps -e

   The names of the processes that are running will appear in the right column of the display. If **obit** and **rlsx** are running, check on their status and restart them if necessary. The instructions for this procedure are detailed in the section "Checking the Status of **obit** and **rlsx**" earlier in this chapter.

2. Your terminal characteristics may have been left in an undefined state. If this is the case, you must reinitialize those characteristics. First, try typing **Ctrl-J** or **Ctrl-M** and then enter an exit command followed by a second **Ctrl-J** or **Ctrl-M**. Even though this command may not be echoed on your display, it may cause the **getty** program to reset your terminal characteristics. If this strategy fails, it may be possible to run a script containing the **stty** commands needed to set your terminal characteristics. As a last resort, you can log in at the root console and terminate the **sh** or **csh** process associated with your terminal. When you log back in at your terminal, the **getty** program will reset your terminal characteristics.

If you had any index files open when Business BASIC terminated, there is also one task you should perform after restarting the interpreter. Use the Business BASIC utility **INDEXVRFY** to verify the structural integrity of those files.

End of Chapter

Chapter 6
System Security

This chapter explains the security features available in Business BASIC on UNIX systems.

AA Accounts

If your interpreter does not allow all programmers to use privileged statements (**STMB**s, **STMC**s, **STME**s, and **STMU**s), the person who takes care of creating profiles for the users on your system can control who is allowed to use these statements. Only programmers whose usernames begin with the letters *AA* have that privilege. For an explanation of what these statements allow you to do, see the manual *Commands, Statements, and Functions in Business BASIC*.

Further, if you are a privileged user, you can mark any of your programs as run only. If you do so, only other privileged users can list those programs. Nonprivileged users can run the programs if they have execute access to the executable files (see the next section), but not otherwise. To make the program in working storage run only, use the command **STMB 16,1** before you save the program. Doing so sets a run-only flag. You clear the flag using the command **STMB 16,0**.

File Access Privileges

By setting the appropriate permission bits for an executable file or a directory, you can exercise a great deal of control over who can access your files and what kind of access those people have. For general information about the permission bits and an explanation of how to set them, see Chapter 1.

The permission bits for an executable file do just what you would expect. Giving the members of your group or others only execute access to a program is equivalent to setting the read-only flag in your program. You are giving group members or others permission to run the program, but not to list it or change it. Read access allows other users to enter or load your program and to list its contents. Write access allows other users to alter your program.

With directories, the permissions bits mean something a little different. Execute permission gives other users permission to move to the directory using the **cd** command. Read access gives members of your group or others the ability to use the **ls** command to list the contents of the directory. Write access gives other users permission to create new files in the directory and to delete existing ones.

End of Chapter

Chapter 7
Porting Applications to a UNIX System

This chapter is divided into two parts. The first part explains how to move your programs and data files from your current system to a UNIX system. The second part explains what changes you may need to make to your programs for them to run in the UNIX environment.

Transferring Files to Your UNIX System

The first two subsections below deal with the general procedure for moving programs and data files from a DG/RDOS or AOS/VS system to a UNIX system. The two main complications you will encounter are (1) that you must move your programs as listing files because Business BASIC for UNIX systems has a new save file format and (2) that the link files that point to the volume label file in a logical file database structure do not function as links on a UNIX system. The third and fourth subsections explain how to use tools provided with your Business BASIC software to convert save files to listing files and vice versa.

NOTE: Since UNIX distinguishes between lowercase and uppercase letters in filenames, it is important to consider how you will name the directories you create on your UNIX system to hold programs and data files, and how you will name the programs and data files themselves. We recommend that you use no lowercase letters in the names of these directories and files. The reasons for this recommendation are discussed in the section "Accessing Files from a Business BASIC Program" in Chapter 2.

Moving Programs to a UNIX System

As mentioned above, the save file format in Business BASIC for UNIX systems is different from the save file format on your current system. Therefore, to move your programs to a UNIX system, you must perform the three steps listed below.

1. If you do not have a listing file (produced with the **LIST** command) for each of your programs, you must start Business BASIC, load each program (using the **LOAD** command), and create a listing file for each program. It is these listing files that you move to your UNIX system.

2. If your programs are stored on an AOS/VS system, you can use the program **ftp** (File Transfer Protocol) to move your files over a TCP/IP network to your UNIX system. It is possible that you may receive the error message `Illegal file type` while transferring a file. If you see this message, create an empty file of type UDF, copy to this new file the contents of the file you were unable to send, and then transfer the UDF file.

WARNING: To prevent corrupting your files while using **ftp**, make sure that you transfer files in binary mode only. You specify binary mode by entering the command **type binary**. Issue this command before you attempt to send or retrieve a file.

If your programs are located on a DG/RDOS system, you must first move the programs to an AOS/VS system and then use **ftp** to move them to your UNIX system. The TCP/IP product, of which **ftp** is a part, is not available on DG/RDOS systems. You can move your programs from a DG/RDOS (not RDOS) system to your AOS/VS system in the following way. Use the DG/RDOS utility **IMOVE** to dump the programs to tapes or diskettes and the AOS/VS CLI command **LOAD** to load the files on your target system. The procedure for moving programs from an RDOS system to an AOS/VS system is different. In this case, dump your programs to a tape, disk, or diskette using the RDOS CLI command **DUMP**. Then load the programs onto your target system using AOS/VS's **RDOS** utility. Use the utility's **LOAD** command to restore programs from an RDOS dump file and the **GET** command to retrieve files from an RDOS disk. If the same type of portable media is not available on your DG/RDOS or RDOS system and your AOS/VS system, you also have the option of transferring files over an asynchronous line using a product such as DG/BLAST.

Regardless of the type of system you are working on currently, if you have a large number of listing files to transfer, you may want to dump those files to a single disk file before running **ftp**. To create such a dump file, use the Business BASIC CLI command **DUMP**. Once you have moved the file to your UNIX system, use the BASIC CLI command **LOAD** to restore your listings.

3. After you have moved your listing files to your UNIX system, start the Business BASIC interpreter on that system, use the **ENTER** command to bring each program into working storage, and then use the **SAVE** command to produce a save file.

If you have a large number of programs to convert to the new save file format, you may want to use the conversion tools discussed later in this chapter.

Moving Data Files to a UNIX System

You move data files to a UNIX system just as you move listings of your programs. See step 2 in the previous section.

The only special thing about data files is that you should not move to your UNIX system any link files that point to a volume label file in a logical file database structure. Standard UNIX System V does not support symbolic link files.

To replace these missing link files once you have transferred the remainder of your logical file database structure, start Business BASIC and then create the necessary link files using the BASIC CLI command **LINK** or the command **STMC 21**. See Chapter 1 for an explanation of the difference between operating-system link files and Business BASIC link files.

Conversion Tools for AOS/VS Users

If you have a large number of Business BASIC save files on your AOS/VS system, the conversion process may become tedious. Remember, you must load and create a listing file for each program on your current system. Then, once you have moved the programs to your UNIX system, you must enter each program into working storage and request that the interpreter produce a save file. If this job seems too imposing, try using the conversion tools **VS_2_UNIX.CLI** and **VS2UNIX.BA**, which were supplied with your Business BASIC software. You can use these tools to automate the conversion process by following the directions below:

1. Before you begin the conversion, back up all of your programs.

2. After you install Business BASIC on your UNIX system, the conversion tools **VS_2_UNIX.CLI** and **VS2UNIX.BA** are located in the directory **CONVERT**, which is subordinate to the directory in which you installed Business BASIC. Use the program **ftp** to move these files to your AOS/VS system.

   WARNING: To prevent corrupting your files while using **ftp**, make sure that you transfer files in binary mode only.

   You can place the program listing **VS2UNIX.BA** and the macro **VS_2_UNIX.CLI** in any directory that will be on your search list when you run the macro, except the directory in which your save files reside. We recommend, however, that you put the program listing in the directory **$SYS**, **$SYSLIB**, or **$SYSLIB3**, and the macro in a directory that is on your current search list, possibly **:UTIL**.

3. Next, you must enter and save the program **VS2UNIX.BA**, so move to the directory that contains the program. Now start a Business BASIC interpreter. If you are running revision 2.xx of Business BASIC, you must log in using an AA account. If you are running a later version, either log in using an AA account or start an interpreter that allows everyone to use privileged statements. Then type the command **ENTER "VS2UNIX.BA** to bring the program into working storage. At this point, you may need to modify the program according to the directions below.

   * If you are running revision 2.xx of Business BASIC, delete lines 2310 and 2320:

     ```
     2310 REM if filetype <> 88 (BBS) return - Rev. 3 and later
     2320 IF ASC(STAT$[2,2])<>88 THEN RETURN
     ```

   * We recommend that, when it is time to run **VS2UNIX**, you execute the **VS_2_UNIX.CLI** macro from a directory that contains nothing but Business BASIC save files and links to Business BASIC save files. If this is not possible, you can indicate to the program which files it is to process by assigning to the string variable TEMPLT$ a template that matches the names of those files and no others. Because of a limitation of the system call the program uses in line 2020, the legal template characters are letters, digits, the asterisk, the minus sign, and the plus sign. The backslash (\) is not legal. If you use it in a template, the program will not process any files.

     After making any necessary changes to the program, save it by typing **SAVE "VS2UNIX**. Then leave Business BASIC by typing **BYE**.

4. You must now make several changes to the macro **VS_2_UNIX.CLI** as detailed below.

   * Change the first line so that it reads `[!EQUAL,1,1]`.

   * Edit the two lines that set your search list for use with Business BASIC. For example, your system may require that the lines read

     ```
     SEA :SYS:BBASIC_5.10<,:$SYSLIB> [!SEA]
     ```

   * Change the two occurrences of the interpreter name **AOSVSBB** to the name of the interpreter you plan to execute.

5. Move to the directory that contains your Business BASIC save files and execute the macro **VS_2_UNIX.CLI** by typing **VS_2_UNIX**. You will see a greeting from Business BASIC and a list of the save files **VS2UNIX** will convert to listing files. The interpreter will then stop itself, and you will see a message indicating that the macro has queued a batch job. When the conversion is complete, AOS/VS will display a message indicating that the job has finished running. The newly created listing files will have the filename extension **.LS**.

   CAUTION: If a file exists whose name consists of the name of a save file in the current directory and a **.LS** extension, the macro deletes that file before it creates the listing file for the save file.

6. At this point, you should look at the following output files: **BB_CNVRT_LOG**, **BB_OUTPRE**, and **BATCHOUTERR**. **BB_CNVRT_LOG** lists the save files that have been processed. **BB_OUTPRE** records any Business BASIC errors that occurred during the conversion process. For example, one of your save files may have been protected so that no listing file was produced for that program, or your batch job may have tried to delete a listing file with the permanence attribute. **BATCHOUTERR** lists any AOS/VS errors that occurred. Do not go on to the next step until you have examined these files.

7. Use **ftp** to move the file **BBIN_UNIX** and the listing files produced in the step above to your UNIX system. Again, to prevent corrupting your files while using **ftp**, make sure that you transfer files in binary mode only.

   If you have a large number of listing files to transfer, you may want to dump them to a single file using the Business BASIC CLI command **DUMP** before the transfer and restore them on your UNIX system using the BASIC CLI command **LOAD**. If you plan to restore your listings using the **LOAD** command, remember that you must execute Business BASIC with the **-c** and **-P** options so that you can use the BASIC CLI.

8. On your UNIX system, move to the directory that contains the file **BBIN_UNIX** and the program listings that you want to convert to save files. Then, issue a command that will start Business BASIC and direct the interpreter to take its input from **BBIN_UNIX**. You may also want to supply the name of a file to which Business BASIC can write its output. For example, if the name of your interpreter is **bbasic**, you might enter the command

   $ bbasic < BBIN_UNIX > errors

   This command will cause the interpreter to perform the following steps for each listing file (*filename*.**LS**) you have imported. First, it checks to see whether a save file named

filename exists in the current directory. If one does exist, the interpreter changes its name to *filename*.**BU**. The interpreter then enters each listing file and saves the program in a file named *filename*.

A Conversion Tool for DG/RDOS Users

As was mentioned earlier, you must export your DG/RDOS Business BASIC programs as listings because the save file format used in UNIX Business BASIC is new. The program **BBCNVRT1.BA**, which is supplied with your Business BASIC release, can help you automate the process of converting DG/RDOS save files to program listings.

After you install Business BASIC on your UNIX system, the program **BBCNVRT1.BA** is located in the directory **CONVERT**, which is subordinate to the directory in which you loaded Business BASIC. Use the program **ftp** to move this file to an AOS/VS system. Then move the file from your AOS/VS system to your DG/RDOS or RDOS system. To move the file to a DG/RDOS system, use the AOS/VS CLI command **DUMP** to dump the file to a tape or diskette and then the DG/RDOS utility **IMOVE** to load the file onto the target system. If your target is an RDOS system, use AOS/VS's **RDOS** utility to put the file on a tape, disk, or diskette and then the RDOS CLI command **LOAD** to load the file onto your RDOS system. For further details about moving files between a DG/RDOS or RDOS system and an AOS/VS system, see step 2 in the section "Moving Programs to a UNIX System."

Once you have moved the conversion program to your DG/RDOS system, read the comments at the beginning of the file. These comments provide the directions you need to run the program.

After converting your save files to listing files, move your listings to your UNIX system using the same commands and programs you used to move **BBCNVRT1.BA** in the opposite direction. You should then produce save files on your UNIX system. To do this, you can start an interpreter, and then manually enter and save each program. Or you can create a text file containing commands similar to the ones shown below and start the interpreter using this file as input.

```
ENTER "PROGRAM1.LS
SAVE "PROGRAM1
NEW
ENTER "PROGRAM2.LS
SAVE "PROGRAM2
NEW
ENTER "PROGRAM3.LS
SAVE "PROGRAM3
NEW
.
.
.
```

For this arrangement to work, all of your listing files and the text file must reside by themselves in the same directory.

If you create such a file, name it **UNIX_INPUT**, and want to use the file as input to an interpreter named **bbasic**, make sure that you are in the directory that contains your listings and then, at the shell prompt, enter the following command:

```
$ bbasic < UNIX_INPUT > errors
```

Business BASIC will convert your listing files to save files and record any errors in the file **errors**.

Making Changes to Your Programs

As far as possible, Business BASIC for UNIX systems is compatible with AOS/VS Business BASIC. However, if you are porting applications from AOS/VS, or from DG/RDOS, to a UNIX system, there are a number of issues you need to be familiar with. The first section below, "Necessary Changes," lists a number of changes you may need to make to your source files regardless of the operating system you are working on currently. The second section covers items that may require change if you are currently working on DG/RDOS, and the third covers items that may be affected if you are working on AOS/VS. The final section compares Business BASIC resource limits on DG/RDOS, AOS/VS, and UNIX systems.

Necessary Changes

Although there is a very high degree of compatibility between AOS/VS Business BASIC and Business BASIC for UNIX systems, and to a lesser extent between DG/RDOS Business BASIC and Business BASIC for UNIX systems, you will have to make some changes to your applications as you move them to the UNIX environment. The areas that may require change regardless of the system you are using currently are listed below and are discussed at length in the succeeding subsections.

- Error handling
- Function keys
- Pathnames
- **PRINT** terminal control characters
- Reserved words
- **STMB** statements
- **STMC** statements
- **SYS** functions
- Terminal command characters

Error Handling

Whether you are coming from a DG/RDOS or AOS/VS environment, you must use a new algorithm to print an error message after trapping an error code with an **ON ERR** statement. The procedure to use is mapped out below.

1. After the error occurs, examine the value of **SYS(40)**, which returns the same value as **SYS(7)**. If the value is positive, the error was a Business BASIC error, one detected by the interpreter. To retrieve the text associated with the error, use the function call **ERM$(SYS(40))**. If the value of **SYS(40)** is negative, proceed to step 2.
2. Check the value of **SYS(41)**, which returns the same value as **SYS(7)** and **SYS(40)**. This value will be a negative number. If the value is not -60, the last error to occur was a DG/RDOS I/O error. To retrieve the text associated with the error, use the function call **ERM$(SYS(41))**. If the value of **SYS(41)** is -60, proceed to step 3.
3. Examine the value of **SYS(42)**, which returns the same value as **SYS(31)**. If its value is not -276, the last error to occur was an AOS/VS I/O error that could not be translated to a DG/RDOS error message. You can retrieve the text associated with the error by using the function call **AERM$(SYS(42))**. If the value of **SYS(42)** is -276, go on to step 4.
4. Check the value of **SYS(43)**. This value will be the code associated with a UNIX error that could not be translated to an AOS/VS error. You can call up the text associated with the error code by using the function call **UERM$(SYS(43))**.

Function Keys

Many existing applications running on DG/RDOS and AOS/VS systems depend on the way that Data General terminals in DG mode handle function keys. These programs assume that pressing a function key will enter a two-byte sequence. The first byte, the function key header, will contain a decimal 30, and the second byte will indicate which function key was pressed. Further, many programs count on the fact that F1, Shift-F1, Ctrl-F1, and Shift-Ctrl-F1 produce different values. Unfortunately, on a UNIX system, where your terminal will not be running in DG mode, pressing a function key may enter more than two bytes; the function key header is probably not a decimal 30; and F1, Shift-F1, Ctrl–F1, Shift-Ctrl-F1 may all produce the same value.

To help you overcome the function-key-header problem, Business BASIC for UNIX systems includes a new function, **SYS(50)**, which returns the value of the function key header. Thus, you can make the function key header the primary interrupt character by entering the command:

```
STMA 4,6,SYS(50)
```

Presently, **SYS(50)** returns 30 on UNIX systems, so the statement **STMA 4,6,30** works also. However, you should not count on **SYS(50)** returning 30 in future revisions of the product.

Business BASIC for UNIX systems also introduces **SYS(51)**, which returns the number of the last function key pressed. That is, if the last key pressed was F3, **SYS(51)** will return a 3.

You should eventually use these new **SYS** functions wherever a program checks to determine whether the user has pressed a function key or to determine which function key the user has pressed. However, for the short term, you can count on the following behavior on UNIX systems. If an **INPUT** statement is executing and the user presses a function key, the interpreter will place a two-byte sequence in a buffer. For example, if the user presses F1, the interpreter will place the values 30 and 113 in the buffer. This arrangement allows existing code that tests for such values to run unchanged.

The fact that on many terminals in use on UNIX systems F1, Shift-F1, Ctrl-F1, and Shift-Ctrl-F1 produce the same value is a more difficult problem to deal with. If your program tests to see whether the user has pressed Shift-F*n*, Ctrl-F*n*, or Shift-Ctrl-F*n*, you may need to rewrite that part of your application.

NOTE: If you are using the TERMINFO packet **at386_bbux** or **vt220_bbux**, a function key and its shifted counterpart produce different values.

Pathnames

Since DG/RDOS and AOS/VS use a colon to separate filenames within a pathname and the UNIX operating system uses a slash for that purpose, you must convert all existing pathnames to their UNIX counterparts as part of your port. You can do this either by editing program listings or by using a pathname-conversion tool provided with Business BASIC for UNIX systems. You must use pathname conversion if you plan to use the Business BASIC CLI.

To use the pathname-conversion tool, include the **-P** option on the command line you use to execute the interpreter. The interpreter will then convert all the AOS/VS-style pathnames in your program that meet the following criteria:

- The pathnames may not contain any slashes, the UNIX filename separator.
- The pathnames may not be preceded by a backslash (\). You can place this character before any pathname that you do not want Business BASIC to convert. If you do not use the **-P** option when you start the interpreter, the backslash has no special meaning to Business BASIC.

In the conversion process, the interpreter makes the following changes:

- Colons are translated to slashes.
- Each caret (^) is translated to ../.
- An equal sign is translated to ./.
- All lowercase letters are translated to uppercase.

Because Business BASIC converts lowercase letters to uppercase, all UNIX directories that are referred to in a converted pathname must have names that contain no lowercase letters.

PRINT Terminal Control Characters

The terminal control characters listed in Table 7–1 are supported on DG/RDOS and AOS/VS systems, but *not* on UNIX systems.

Table 7–1 Unsupported PRINT Terminal Control Characters

Character	What It Does
–26	Moves cursor to next tab stop
–31	Clears unprotected screen positions
–34	Locks keyboard
–35	Unlocks keyboard
–44	Sets program mode
–45	Clears program mode
–46	Sets block mode
–47	Clears block mode
–48	Sets flag 1
–49	Clears flag 1
–50	Sends line (unprotected fields)
–51	Sends line (all fields)

Reserved Words

None of the reserved words listed in the file **APERM.PS** (which is located in **/usr/opt/bbux/SYSLIB**) can be used as a variable name. In DG/RDOS and AOS/VS Business BASIC, this restriction is not enforced, so your current programs may violate this rule. However, those programs will not run on UNIX Business BASIC until you change the names of the variables whose names appear in this file.

STMB Statements

- In DG/RDOS and AOS/VS Business BASIC, **STMB 0** returns addresses for a number of internal data structures. Almost all of these internal structures have been changed completely or removed in Business BASIC for UNIX. Therefore, only **STMB 0,21** is supported in the UNIX environment. All other items return 65,535.
- The User Status Table does not exist in UNIX Business BASIC. Therefore, any program that reads information from the User Status Table using such system calls such as **STMB 1,1,14** and **STMB 5,2,14** will no longer work. In addition, code which uses **STMB**s to store information in memory will not work if the memory locations are calculated based on information obtained from the User Status Table.

STMC Statements

Neither of the two STMC statements listed below will function properly in a UNIX Business BASIC program:

- Under DG/RDOS, **STMC 14** determines whether a foreground program is running, and under AOS/VS, the statement determines whether a process has a son. Under UNIX **STMC 14** always returns a 0 because a UNIX process does not know whether it has children on not.
- In DG/RDOS and AOS/VS Business BASIC, **STMC 38** opens a channel to a magnetic tape drive. The statement is illegal in UNIX Business BASIC.

SYS Functions

Programs containing the functions **SYS(4)** and **SYS(30)** may require changes as well.

In DG/RDOS **SYS(4)** uniquely identifies the console at which you are working. In AOS/VS **SYS(4)** returns a console number, and **SYS(30)** indicates whether the console is a virtual console. On UNIX systems, several devices may have the same unit number, for example, **tty01**, **ttyp01**, and **vt01**. To identify the UNIX device, you use **SYS(4)** to return the unit number and a new function, **SYS(33)**, to determine the device-type number. Of course, if all the users on your system work on the same type of device, **SYS(4)** alone will distinguish between them.

In addition, if you have a program that uses **SYS(30)** to determine which operating system it is running on, you will probably want to test bit 12 of the word whose value is returned by this function to determine whether the program is running on a UNIX system.

Terminal Command Characters

In the DG/RDOS and AOS/VS environments, Business BASIC allows you to embed DG mode terminal command characters in string literals. When the interpreter sends these strings to the display, not only is the string printed, but a terminal command is executed. This may be a command to move the cursor to a new line, to turn reverse video on, and so on.

For example, a program may contain the following line:

```
* 10 PRINT "<12>This is the top of the screen.<7>"
```

When run under DG/RDOS and AOS/VS, this code clears the screen (<12> represents a form feed), prints the message on the first line of the screen, and sounds the terminal's bell (<7> represents the bell).

On UNIX systems, Business BASIC supports a subset of the DG mode terminal commands, if you use the **-C** option on your Business BASIC command line. If you do not use that option, Business BASIC prints a representation of each control character in the string. That is, if you were to enter the program line shown above, the output would be

```
* 10 PRINT "<12>This is the top of the screen.<7>"
* RUN
^LThis is the top of the screen.^G

*
```

The subset of commands Business BASIC supports is shown in Table 7–2 below:

Table 7–2 Supported Terminal Control Characters

Control Code in Decimal	Command Name
<7>	Bell
<8>	Window home
<10>	New line
<12>	Erase window
<13>	Carriage return
<14>	Blink on
<15>	Blink off
<16><col.><row>	Position cursor in window
<20>	Underscore on
<21>	Underscore off
<23>	Cursor up
<24>	Cursor right
<25>	Cursor left
<26>	Cursor down
<28>	Dim on
<29>	Dim off
<30><68>	Reverse video on
<30><69>	Reverse video off

In addition, when you execute Business BASIC with the **-C** option, the interpreter discards the following commands when they appear in string literals:

- <1>, the "print form" command
- <3>, the "blink enable" command
- <4>, the "blink disable" command

The interpreter handles all other commands the same way it handles all commands when you do not use the **-C** option.

A word of warning. Even though you can use the terminal control commands shown in Table 7-2 in any Business BASIC program, you will produce more generic, portable code if you use **PRINT @** functions in their place. For instance, you should recode the program line shown earlier in this way:

```
* PRINT @(-30);"This is the top of the screen.";@(-25)
```

DG/RDOS-Only Items

If you are porting an application from DG/RDOS to a UNIX system, check the following list for possible sources of incompatibility:

STMA Statements

Many of the **STMA** statements are valid only in DG/RDOS Business BASIC programs. You must revise any program that uses an **STMA** statement to perform one of the following tasks:

- To return or set the value of a detach key. (**STMA 3,0** and **STMA 4,0**)
- To determine whether Ctrl-S and Ctrl-Q are enabled on a DG/RDOS multiplexor line, or to disable or enable those control characters. (**STMA 5,11**; **STMA 6,11**; and **STMA 7,11**)
- To detach or attach a job. (**STMA 16** and **STMA 17**)
- To examine, assign, or free a reserved file or device. (**STMA 18**)

STMB Statements

As with the **STMA** statements, many **STMB** statements are legal only in DG/RDOS Business BASIC programs. You must rework any program that uses an **STMB** statement to perform one of the following tasks:

- To reset the **ON IKEY** condition for a job, clear the "ignore IKEY" flag, and then simulate an interrupt, stopping the job. (**STMB 8**)
- To scan the job table for the first available job and execute the **HELLO** program for that job. (**STMB 12**)

- To place a string in the input buffer for a job. (**STMB 13**)
- To set a job's alternate IKEY or unpend flag to 0 or 1 and simulate an interrupt. (**STMB 14**)
- To set the characteristics for a multiplexor line. (**STMB 15**)
- To return the value of a word in the operating system's address space. (**STMB 18**)
- To set the modem status on a multiplexor line. (**STMB 19**)

STMC Statements

Many of the **STMC** statements are valid only in DG/RDOS Business BASIC programs. You must revise any program that uses an **STMC** statement to perform one of the following tasks:

- To change the attributes of a file or a link file. (**STMC 2** and **STMC 3**)
- To retrieve a copy of the user file descriptor for a file opened on a specified channel. (**STMC 4, STMC 33,** and **STMC 51**)
- To assign a temporary name to a disk or tape unit. (**STMC 10**)
- To return the attributes of a file opened on a specified channel. (**STMC 18**)
- To return the name of the current operating system. (**STMC 19**)
- To initialize a device. (**STMC 20**)
- To return the name of the current master directory. (**STMC 22**)
- To disable or enable console interrupts. (**STMC 23** and **STMC 24**)
- To release an initialized device. (**STMC 27**)
- To set the system date. (**STMC 29**)
- To disable, enable, or kill spooling for a device. (**STMC 30, STMC 31,** and **STMC 32**)
- To set the time. (**STMC 34**)
- To create a file and set the date last accessed and the date and time of creation. (**STMC 45, STMC 46,** and **STMC 47**)
- To find the amount of memory allocated to the current ground. (**STMC 48**)
- To close an opened file. (**STMC 53**)

STMD Statements

The **STMD** statements, which allow a privileged user to send messages to and receive responses from any user's terminal, are legal only in DG/RDOS Business BASIC programs.

AOS/VS-Only Items

You need check the following items only if you are porting an AOS/VS Business BASIC program to UNIX Business BASIC.

FKT Characteristic

Some AOS/VS Business BASIC programs depend on the characteristic **FKT** being turned on. This characteristic permits function keys to serve as delimiters in data-sensitive read operations. (You set this characteristic using the AOS/VS CLI command **CHARACTERISTICS/FKT**.) These programs will not work properly on UNIX systems since the latter operating system does not offer a similar characteristic. You can solve this problem in one of two ways. First, you can add to your program the statement **STMA 4,4,SYS(50)** or **STMA 4,5,SYS(50)**. The first of these statements causes the function key header to become the primary unpend key, and the second causes the function key header to become the secondary unpend key. Second, you can add the statement **PRINT @(-5,SYS(50))** or **PRINT @(-4,SYS(50))** to your program. These statements perform the same functions as the **STMA** statements just mentioned.

INFOS II Statements

Since only AOS/VS Business BASIC allows you to work with INFOS II files, none of the INFOS II statements are legal in the UNIX environment.

STMB Statement

STMB 24, which allows you to perform an AOS/VS system call, works only in the AOS/VS environment.

STME Statements

Many of the **STME** statements are valid only in AOS/VS Business BASIC programs. You must revise any program that uses an **STME** statement to perform one of the following tasks:

- To access the initial IPC message that the CLI sends when it creates a new process (**STME 3**). You can replace this statement with **STMU 4**.
- To determine the specifications set for a file when it was created (**STME 5**, **STME 6**, and **STME 7**). You can replace these statements with **STMU 0**, **STMU 1**, and **STMU 2** respectively.
- To create a file (**STME 8** and **STME 9**). You can replace these statements with **STMU 3**.
- To return the device characteristics of **@INPUT** or **@OUTPUT**, or to set the characteristics of **@INPUT**. (**STME 12**, **STME 13**, and **STME 14**)
- To send a messages to, or receive a message from, another process. (**STME 15**, **STME 16**, **STME 20**, **STME 21**, and **STME 22**)
- To create an IPC port, find the owner of a global port, look up a port number, or translate a port number. (**STME 23**, **STME 24**, **STME 25**, and **STME 26**)

SYS(9)

On AOS/VS and UNIX systems, **SYS(9)** returns your process's PID. On AOS/VS systems, this identification number will be a maximum of four digits. However, under UNIX this number is often five digits long. If you have coded a program so that it allows **SYS(9)** to return only four or fewer digits, the program may not work correctly.

Resource Limits

While you are making changes to your applications, you may want to take advantage of some of the expanded resource limits available in UNIX Business BASIC. These limits are discussed in Chapter 2 and are summarized in Table 7–3 below so that you can compare them with the limits on your current system:

Table 7–3 Resource Limits in DG/RDOS, AOS/VS, and UNIX Business BASIC

Limit	UNIX Systems	AOS/VS	DG/RDOS
Highest line number	99,999	32,767	32,767
Line length	256 chars.	256 chars.	132 chars.
Program size	512 Kbytes	256 Kbytes	22–38 Kbytes
Number of variables	8,192	348	348
Length of variable name	32 chars.	6 chars.	6 chars.
Dimensions in a numeric array	8	2	2
Dimensions in a string array	8	N/A	N/A
Length of filename	254 chars. 13 on 386/ix	31 chars.	13 chars.
Number of open files per user	100[1]	150	16
Size of common area	2,048 bytes	512 bytes	512 bytes
DEF levels	26	4	4
FOR/NEXT levels	32	8	8
DO/WHILE/UNTIL levels	32	N/A	N/A
GOSUB levels	32	8	8

[1] If you reconfigure the UNIX kernel as recommended in the Business BASIC Release Notice.

End of Chapter

Appendix A
Error Messages

This appendix lists the valid error codes that may be returned by the functions **SYS(40)**, **SYS(41)**, **SYS(42)**, and, **SYS(43)**. It also lists the error messages that you can retrieve with the functions **ERM$**, **AERM$**, and **UERM$**.

When an error occurs, you should use the procedure outlined below to determine the error code and error message associated with that error.

1. Examine the value of **SYS(40)**. If the value is positive, the error was a Business BASIC error, one detected by the interpreter. The codes for this type of error are cataloged in Tables A–1 and A–2. To retrieve the text associated with the error, use the function call **ERM$(SYS(40))**. If the value of **SYS(40)** is negative, proceed to step 2.
2. Check the value of **SYS(41)**. This value will be a negative number. If the value is not –60, the last error to occur was one of the so-called I/O errors listed in Tables 3 through 5. To retrieve the text associated with the error, use the function call **ERM$(SYS(41))**. If the value of **SYS(41)** is –60, proceed to step 3.
3. Examine the value of **SYS(42)**. If its value is not –276, the last error to occur was an AOS/VS I/O error that could not be translated to a DG/RDOS error. Errors of this type are listed in Table A–6. You can retrieve the text associated with the error by using the function call **AERM$(SYS(42))**. If the value of **SYS(42)** is –276, go on to step 4.
9. Check the value of **SYS(43)**. This value will be the code associated with a UNIX error that could not be translated to an AOS/VS error. The UNIX errors for 386/ix systems are listed in Table A–7, and those for DG/UX systems are listed in Table A–8. You can call up the text associated with the error code by using the function call **UERM$(SYS(43))**.

Table A-1 Business BASIC Syntax Error Messages

Record number in BASIC.ER	Error Code	Error Message
1	1	Illegal character
2	2	Statement or command syntax is invalid
3	3	READ/DATA types inconsistent
4	4	SYSTEM (program is lost)
5	5	Statement number
6	6	Excessive number of variables
7	7	Statement may not be used as a command
8	8	Specification
9	9	Illegal reserved file name
10	10	Reserved file in use
11	11	Parenthesis
12	12	Illegal command
13	13	Line number
14	14	No more program memory
15	15	End of DATA
16	16	Arithmetic
17	17	Unassigned variable
18	18	GOSUB nesting
19	19	RETURN - no GOSUB
20	20	FOR nesting
21	21	FOR - no NEXT
22	22	NEXT - no FOR
23	23	No more data memory
24	24	No available channels
25	25	Option is not in this system
26	26	Program/data overflow
27	27	Illegal file number
28	28	New dimension exceeds old dimension
29	29	Expression
30	30	Illegal mode
31	31	Subscript
32	32	Undefined function
33	33	Function nesting
34	34	Function argument
35	35	Illegal format string
36	36	String size
37	37	User routine
38	38	Undimensioned string

(continued)

Table A-1 Business BASIC Syntax Error Messages

Record number in BASIC.ER	Error Code	Error Message
39	39	Duplicate matrix
40	40	Matrices sizes
41	41	Matrix DIM
42	42	File already opened
43	43	Matrix not square
44	44	File unopened
45	45	Illegal record length
46	46	Input invalid
47	47	Wrong mode
48	48	Not a save file
49	49	No room for directory
50	50	Invalid operator command
51	51	User not on system
52	52	User in NOMSG state
53	53	Renumbering error(s)
54	54	Statement length
55	55	Block I/O error
56	56	Attempt to LOCK same area twice
57	57	Data base record is LOCKed
58	58	Incompatible save file
59	59	Zero STEP
60	60	Line too long
61	61	Missing right parenthesis in format
62	62	Save file is run only
63	63	Incorrect number of arguments
64	64	Incorrect argument type
65	65	Program swapping error
66	66	String argument in error
67	67	Error in index file or index parameters
68	68	Index file full
69	69	Excessive IPB errors
70	70	Invalid job/terminal number
71	71	Job/terminal already attached
72	72	Job is not waiting on input
73	73	Edit buffer is empty
75	75	Program may not be listed
76	76	Non-fatal system error
77	77	Illegal record number

(continued)

Table A-1 Business BASIC Syntax Error Messages

Record number in BASIC.ER	Error Code	Error Message
78	78	MSG in progress
79	79	Undefined RLS2 function code
80	80	Error executing ON ERR statement
81	81	Illegal record status
82	82	Data file full
83	83	Error from INFOS II
84	84	Not a data base file
85	85	Illegal command with optimized file
86	86	LFTABL$ string error
87	87	Missing ELSE or END IF
88	88	Extra ELSE or missing END IF
89	89	Illegal file type
90	90	Logical file does not exist
91	91	Attempt to issue LOCK/UNLOCK without lock server running
92	92	Parameter range error
93	93	Maximum number of locks has been exceeded
94	94	Incompatible revision of volume label file
95	95	Maximum number of duplicate keys exceeded
96	96	DO with no matching END LOOP, WHILE or UNTIL
97	97	END LOOP, WHILE or UNTIL with no matching DO
98	98	DO with conditional at top and bottom
99	99	DO nesting
100	100	Missing shared page server environment variable
101	101	Non-existent shared page directory
102	102	Shared page server not running where expected
103	103	Can't create message queue for server responses
104	104	Can't receive responses from server
105	105	Can't send request to shared page server
106	106	Variable in use
107	107	Illegal name change - data types inconsistent

(concluded)

Table A-2 Utility Program Error Messages

Record number in BASIC.ER	Error Code	Error Message
128	128	Invalid or out-of-range field
129	129	No key was given
130	130	Record does not exist
131	131	Record already exists
132	132	Validation error, key was changed
133	133	Function is invalid without a FIND function first
134	134	Control file is in error
135	135	Invalid format or page
136	136	No next record
137	137	Index for this record may be all messed up
138	138	Key could not be deleted
139	139	Request not completed
140	140	Invalid account/password
141	141	INPUT timed out
142	142	Device is assigned
143	143	Illegal function
144	144	File not found
146	146	Key already exists and duplicates not allowed
147	147	Insufficient space for logical file
148	148	File not on sector boundary
149	149	Record out of sequence
150	150	Illegal blocking factor
151	151	Illegal key length
152	152	Data dictionary does not exist
153	153	File types do not match
154	154	Wrong program

Table A-3 DG/RDOS System Error Messages

Record number in BASIC.ER	Error Code	Error Message
256	0	Illegal channel number
257	-1	Illegal file name
258	-2	Illegal system command
259	-3	Illegal command for device
260	-4	Not a save file
261	-5	File already exists
262	-6	End of file
263	-7	File read protected
264	-8	File write protected
265	-9	File already exists
266	-10	File does not exist
267	-11	Permanent file
268	-12	File attribute protected
269	-13	File not open
270	-14	Fatal utility error
271	-15	Execute CLI.CM (no error)
272	-16	Invisible error code
273	-17	Channel already in use
274	-18	Line too long
275	-19	Attempt to restore a non-existent image
276	-20	Parity error
277	-21	Push depth exceeded
278	-22	Insufficient memory to execute program
279	-23	File space exhausted
280	-24	File data error
281	-25	Unit improperly selected
282	-26	Illegal starting address
283	-27	Attempt to read into system space
284	-28	File accessible by block I/O only
285	-29	Files on different directories
286	-30	Device not in system
287	-31	Illegal overlay number
288	-32	File not accessible by block I/O
289	-33	Invalid time or date
290	-34	Out of TCB's
291	-35	Signal to busy address
292	-36	File already squashed error
293	-37	Device already in system

(continued)

Table A-3 DG/RDOS System Error Messages

Record number in BASIC.ER	Error Code	Error Message
294	-38	Insufficient contiguous blocks
295	-39	Simultaneous read or write to mux line
296	-40	Error in user task queue table
297	-41	No room for directory
298	-42	Illegal directory specifier
299	-43	Unknown directory specifier
300	-44	Partition too small
301	-45	Directory depth exceeded
302	-46	Directory in use
303	-47	Link depth exceeded
304	-48	File in use
305	-49	Task ID error
306	-50	Common size error
307	-51	Common usage error
308	-52	File position error
309	-53	Insufficient room in data channel map
310	-54	Directory not initialized
311	-55	No default directory
312	-56	Foreground already active
313	-57	Error in partition set
314	-58	Directory shared
315	-59	No room for UFT's
316	-60	Address error on .SYSTM argument
317	-61	Not a link entry
318	-62	Program not checkpointable
319	-63	SYS.DR error
320	-64	MAP.DR error
321	-65	Device timeout
322	-66	Entry not accessible by a link
323	-67	Not a source file
324	-68	Transmission terminated by receiver
325	-69	System deadlock
326	-70	I/O terminated by channel close
327	-71	Spool files active
328	-72	Task not found for abort
329	-73	Device previously opened
330	-74	System stack overflow
331	-75	No MCA receive request outstanding

(continued)

Table A-3 DG/RDOS System Error Messages

Record number in BASIC.ER	Error Code	Error Message
332	-76	Attempt to release an open device
333	-77	.XMT or .IXMT messages must be non-zero
334	-78	'You can't do that'
335	-79	.TOVLD not loaded for queued overlay tasks
336	-80	Operator messages not SYSGEN'ed
337	-81	Disk format error
338	-82	Disk has invalid bad block table
339	-83	Insufficient space in bad block pool (core)
340	-84	Attempt to create a zero length contiguous file
341	-85	Program is not swappable
342	-86	Blank tape
343	-87	ALM line not ready
344	-88	Console interrupt received
345	-89	Read overrun error
346	-90	Read framing error
347	-91	Too many soft errors
348	-92	QTY buffer overflow

(concluded)

Table A-4 Business BASIC I/O Error Messages

Record number in BASIC.ER	Error Code	Error Message
383	-127	Device is assigned, but not to you
384	-128	Illegal function
385	-129	Variable length blocks not allowed
386	-130	Rewrite mode not allowed for non-disk device
387	-131	Illegal function for device
388	-132	Open processing error
389	-133	Function does not process specified record format
390	-134	File already in use
391	-135	File currently opened for exclusive use
392	-136	File not open
393	-137	Peripheral device currently opened
394	-138	VL file processing error
395	-139	Unresolved resource conflict
396	-140	Rewrite inverted with changed data address
397	-141	File name already in use
398	-142	Block size exceeds window
399	-143	Virtual memory exhausted
400	-144	System translation table load error
401	-145	The system cannot open the file as requested
402	-146	Volume close error
403	-147	Insufficient space to open file
404	-148	Beyond logical bounds of file
405	-149	Translation specification error
406	-150	Volume does not exist
407	-151	Record is locked and 'hold' not requested
408	-152	Disk space exhausted
409	-153	Record beyond bounds of file
410	-154	Error while closing volume during internal volume switch
411	-155	Physical I/O error
412	-156	Residual disk error
413	-157	Disk or magnetic tape timeout
414	-158	Illegal access method
415	-159	Insufficient parameters for translation
416	-160	Special file open error in preopen
417	-161	Special file close error in preopen
418	-162	Internal special file close error
419	-163	RDOS open failure
420	-164	Volume already exists

(continued)

Table A-4 Business BASIC I/O Error Messages

Record number in BASIC.ER	Error Code	Error Message
421	-165	Zero length transfer request on disk device
422	-166	ISAM update conflict
423	-167	Index naming error
424	-168	Index not in data base index definition file
425	-169	File name too long
426	-170	No node space available
427	-171	Mag tape processing error
428	-172	System does not support device
429	-173	End of volume for output file
430	-174	End of volume for input file
431	-175	ISAM comparison error
432	-176	ISAM resolution error
433	-177	Illegal relative motion
434	-178	Illegal node address encountered internally
435	-179	Invalid entry address encountered internally
436	-180	Top level error
437	-181	Subindices not allowed
438	-182	Subindex not present
439	-183	(sub)index boundary encountered
440	-184	Delete positioning error
441	-185	Multiple key write error
442	-186	Key too long or of zero length
443	-187	Invalid entry number encountered internally
444	-188	Neither 'keyed' nor 'relative motion' specified
445	-189	Key already exists and duplicates not allowed
446	-190	Key positioning error
447	-191	Illegal record length
473	-217	Illegal variable
474	-218	Illegal text argument
475	-219	Text argument too long
512	-256	No data base record for index entry
513	-257	Node size too big for page
514	-258	Node size will not hold 3 entries
515	-259	Data base record is locked
516	-260	Subindex in use encountered during processing
517	-261	File and system versions incompatible
518	-262	Too many subindex links
519	-263	Subindex already present

(continued)

Table A-4 Business BASIC I/O Error Messages

Record number in BASIC.ER	Error Code	Error Message
520	-264	Subindex level limit exceeded
521	-265	Index entry with subindex may not be deleted
522	-266	Physical delete access must be 'keyed'
523	-267	Index entry is locked
524	-268	Write access must be keyed
525	-269	Illegal label format on mag tape
526	-270	Illegal label specification
527	-271	Volume identifiers do not match
528	-272	File identifiers do not match (internal error)
529	-273	File sequence numbers do not match (internal error)
530	-274	File generation numbers do not match (internal error)
531	-275	File expiration date not yet reached
532	-276	Block count on trailer label disagrees with actual count
533	-277	Record formats do not match
534	-278	Incorrect file section number on header label
535	-279	Excessive position labels
536	-280	System size load error
537	-281	Tape file not found
538	-282	Block size less than 8 bytes
539	-283	Record plus overhead is greater than block size
540	-284	Write is not at end-of-file for shared SAM update file
541	-285	Only one user may write to a shared SAM update file
542	-286	Spooling enabled on illegal device
543	-287	INFOS retrieve key error
544	-288	Delete index positioning error
545	-289	Space management inconsistency
546	-290	Error searching current position table
547	-291	Tape mount cancelled by operator

(concluded)

Table A–5 Business BASIC CLI Error Messages

Record number in BASIC.ER	Error Code	Error Message
448	–192	Not enough arguments
449	–193	Illegal attribute
450	–194	No debug address
451	–195	Command line too long
452	–196	No starting address
453	–197	Checksum error
454	–198	No source file specified
455	–199	Not a command
456	–200	Illegal block type
457	–201	No files match specifier
458	–202	Phase error
459	–203	Too many arguments
460	–204	Too many active devices
461	–205	Illegal numeric argument
462	–206	Fatal system utility error
463	–207	Illegal argument
464	–208	Improper or malicious input
465	–209	Too many levels of indirect files
466	–210	Syntax error
467	–211	Bracket error
468	–212	Parenthesis error
469	–213	Unmatched <>
470	–214	Illegal nesting of <> and ()
471	–215	Illegal indirect filename
472	–216	Illegal nesting of () and {}
473	–217	Illegal variable
474	–218	Illegal text argument
475	–219	Text argument too long

Table A-6 AOS/VS System Error Messages

Error Code	Error Message
-1	Illegal system command
-2	Channel not open
-3	Channel already open
-4	Shared I/O request not map slot aligned
-5	Insufficient memory available
-6	Illegal starting address
-7	Illegal overlay number
-8	Illegal set time argument
-9	No task control block available
-10	?XMT to address already in use
-11	Error in user task queue table
-12	Task ID error
-13	Data channel map is full
-14	System call parameter address error
-15	Task not found for abort
-16	Insufficient room in buffer
-17	File space exhausted
-18	User stack fault
-19	Directory does not exist
-20	Illegal filename character
-21	File does not exist
-22	Filename already exists
-23	Non-directory argument in pathname
-24	End of file
-25	Directory delete error
-26	Write access denied
-27	Read access denied
-28	Append and/or write access denied
-29	No free channels
-30	Release of non-active shared slot
-31	Illegal process priority
-32	Illegal maximum process size
-33	Illegal process type
-34	Console device specification error
-35	Out of swap file room
-36	Device already in system
-37	Illegal device code
-38	Error on shared partition set

(continued)

Table A-6 AOS/VS System Error Messages

Error Code	Error Message
-39	Error on remap call
-40	Illegal agent gate call
-41	No PID available for this process
-42	IPC message longer than buffer
-43	Invalid port number
-44	No matching send
-45	No outstanding receive
-46	Illegal origin port
-47	Illegal destination port
-48	Invalid shared library reference
-49	Illegal record length specified
-50	Attempt to release console device
-51	Device already in use
-52	Attempt to release unassigned device
-53	Attempt to close unopen channel or device
-54	I/O terminated by CLOSE
-55	Line too long
-56	Parity error
-57	Resident process tried to create son and block
-58	Not a directory
-59	Shared I/O request not to shared area
-60	Too many subordinate processes
-61	File read error
-62	Device timeout
-63	Wrong I/O type for OPEN type
-64	Filename too long
-65	Positioning before beginning of file
-66	Caller not privileged for this action
-67	Simultaneous requests on same channel
-68	Illegal file type
-69	Insufficient room in directory
-70	Illegal OPEN
-71	Attempt to access process not in hierarchy
-72	Attempt to block unblockable process
-73	Invalid system call parameter
-74	Attempt to start multiple agents
-75	Channel in use
-76	Not enough contiguous disk blocks

(continued)

Table A-6 AOS/VS System Error Messages

Error Code	Error Message
-77	Stack overflow
-78	Inconsistent bitmap data
-79	Illegal block size for device
-80	Attempt to ?XMT illegal message
-81	Physical unit failure
-82	Physical write lock
-83	Physical unit off-line
-84	Illegal OPEN option for file type
-85	Too many or too few device names
-86	Disk and file system revision numbers don't match
-87	Inconsistent device information block (DIB) data
-88	Inconsistent logical disk unit (LDU)
-89	Incomplete logical disk unit (LDU)
-90	Illegal device name type
-91	Illogical process address space definition
-92	LDU in use, cannot release
-93	Too many directories in search list
-94	Cannot get IPC data from father
-95	Illegal library number given
-96	Illegal record format
-97	Too many or too few arguments to PMGR
-98	Illegal ?GTMES parameters
-99	Illegal CLI message
-100	Message receive disabled by CHARACTERISTICS/NRM
-101	Not a console device
-102	Attempt to exceed maximum index level
-103	Illegal channel
-104	No receiver waiting
-105	Short receive request
-106	Transmitter inoperative
-107	Illegal username
-108	Illegal link number
-109	Disk positioning error
-110	Message text longer than specified
-111	Short transmission
-112	Illogical histogram call
-113	Illegal retry value
-114	ASSIGN error - already your device

(continued)

Table A-6 AOS/VS System Error Messages

Error Code	Error Message
-115	Mag tape request past logical end of tape
-116	Packet specifies stack too small
-117	Packet would create too many tasks
-118	Spooler open retry count exceeded
-119	Illegal ACL
-120	?STMAP buffer contains invalid or write-protected page
-121	Partner process has not opened IPC file
-122	FPU hardware not installed
-123	Illegal process name
-124	Process name already in use
-125	Disconnect error on modem
-126	Process must block to pass generic files
-127	System not installed
-128	Maximum directory tree depth exceeded
-129	Releasing out-of-use overlay
-130	Resource deadlock
-131	File is open, cannot exclusively open
-132	File is exclusively opened, cannot open
-133	Initialization privilege denied
-134	Multiple ?IMSG calls to same DCT
-135	Illegal link
-136	Illegal DUMP format
-137	EXEC not available
-138	EXEC request function unknown
-139	EXEC's process subtree only
-140	Request refused by system operator
-141	Cannot dismount, was not mounted
-142	Switch or argument value greater than 65535
-143	Input file does not exist
-144	Output file does not exist
-145	LIST file does not exist
-146	DATA file does not exist
-147	Recursive generic file OPEN failure
-148	No message waiting
-149	User data area (UDA) does not exist
-150	Illegal device type from VSGEN
-151	AOS/VS restart of system call
-152	Fatal user runtime error

(continued)

Table A-6 AOS/VS System Error Messages

Error Code	Error Message
-153	User commercial stack fault
-154	User floating point stack fault
-155	User data area (UDA) already exists
-156	Illegal screenedit request (PMGR)
-157	?SEND destination device held by ^S (CTRL-S)
-158	Data overrun error
-159	Control point directory max size exceeded
-160	System or bootstrap disk not part of master logical disk (LDU)
-161	Universal system, you can't do that
-162	Execute access denied
-163	Cannot initialize LDU; run FIXUP on it
-164	File access denied
-165	Directory access denied
-166	Attempt to define more than one INFOS II process
-167	INFOS II process not running
-168	Attempt to issue MCA request while direct I/O in progress
-169	Attempt to issue MCA direct I/O with request outstanding
-170	Last task was killed
-171	Resource load or release failure
-172	Zero length filename specified
-173	Buffer overflow
-174	Transmission failure (too many NAKS)
-175	Transmission failure (timeouts)
-176	Disconnect occurred on sync line
-177	EOT character received
-178	Possible lost data on HASP line
-179	Sync DCU inoperative (I/O aborted)
-180	Conversational reply received
-181	End of polling list reached without a response
-182	Illegal relative terminal number
-183	Reverse interrupt response received
-184	Illegal line number specified
-185	Not enough space for polling list
-186	Contention situation while bidding
-187	Out-of-sequence gen entry during SINIT
-188	Request to a non-synchronous line
-189	Not enough memory for communications buffer
-190	Line already enabled when ?SEBL call issued

(continued)

Table A-6 AOS/VS System Error Messages

Error Code	Error Message
-191	Line already disabled when ?SDBL call issued
-192	I/O request on a disabled line
-193	Line in session on initial I/O request
-194	Line not in session on continue I/O request
-195	Buffer byte count larger than system buffer
-196	Bid error (too many NAKS)
-197	Wait-A-Bit received (HASP line only)
-198	User buffer byte pointer invalid
-199	Retry count exceeded
-200	ETX code received
-201	Input status format error
-202	Failure to connect error
-203	Uninterpretable response received
-204	ENQ received
-205	Block check error
-206	Initialization parameter error
-207	Transmitter failure error
-208	Line not multidrop
-209	Not a control station
-210	Polling list not defined
-211	Inconsistent tab format
-212	Cannot delete permanent file
-213	System call abort
-214	Extended context already defined
-215	Unreadable tape or diskette label
-216	Incorrect labeled volume mounted
-217	Incorrect labeled tape fileset
-218	Incorrect file section number
-219	Incorrect labeled tape file generation number
-220	Incorrect labeled tape file version number
-221	No operator available
-222	Unknown tape label format
-223	Extended context already initialized
-224	Extended context not initialized
-225	Extended context not defined
-226	Memory release error
-227	Illegal translation parameter
-228	Missing or invalid argument

(continued)

Table A-6 AOS/VS System Error Messages

Error Code	Error Message
-229	Not in CLI format
-230	Illegal bias factor
-231	CPU time limit exceeded
-232	Error in setting max CPU limit
-233	File element size not multiple of 4
-234	WACK response received
-235	Process is not a server
-236	Connection does not exist
-237	Connection table full
-238	Directory in use - cannot delete
-239	Attempt to grow a shared file
-240	Illegal directory name specification
-241	Network not available
-242	Host already exists
-243	Illegal host specification
-244	Host does not exist
-245	Cannot rename host entries
-246	Empty mailbox on ?RECNW
-247	Unable to access network in this manner
-248	Attempt to create multiple local host
-249	Not awaiting ?IWKUP
-250	Illegal remote ?PROC parameter(s)
-251	Illegal host name
-252	Not proper for a virtual circuit
-253	HDLC - invalid window size
-254	Invalid frame size
-255	SEND active
-256	Invalid call type
-257	Remote is disconnecting
-258	Local received invalid response
-259	Local received CMDR
-260	Local is in CAN'T SEND condition
-261	Local is disconnecting
-262	Local was reset
-263	Buffer overflow
-264	Receive active
-265	Initialization failed
-266	Local received invalid command

(continued)

Table A-6 AOS/VS System Error Messages

Error Code	Error Message
-267	Non-HDLC enable attempted
-268	Interrupt wait task already defined
-269	Map slot error
-270	Get buffer error
-271	Sync DCU inoperative
-272	Error opening SLDCU.PR
-273	Error reading SLDCU.PR
-274	Error closing SLDCU.PR
-275	Error getting memory
-276	Unknown error
-277	Connection has been broken
-278	Attempt HDLC call without DCU 200
-279	Cannot connect to self
-280	No connection
-281	Tape controller does not support this density mode
-282	Indecipherable tape density
-283	File/tape density mismatch
-284	Illegal label level
-285	Illegal label character
-286	Incorrect labeled tape sequence number
-287	Incorrect labeled tape block count
-288	Volid too long or null
-289	Owner ID too long
-290	User labels too long or too many
-291	Record length exceeds block length
-292	AP already busy
-293	AP does not exist
-294	AP WCS is not loaded
-295	Attempt to release a non-AP page
-296	Attempt to release an AP page
-297	Only one file on ANSI level 1 labeled tape
-298	Cannot delete unexpired file on labeled medium
-299	Process console does not exist
-300	Histogram target process termination
-301	Error detected while histogram in progress
-302	Invalid IPC message
-303	Multiple users of file, cannot truncate
-304	Shared file, cannot truncate

(continued)

Table A-6 AOS/VS System Error Messages

Error Code	Error Message
-305	File being truncated, cannot open
-306	Framing error
-307	Internal directory inconsistency
-308	Commercial fault
-309	Fixed point fault
-310	Floating point fault
-311	UNKNOWN MESSAGE CODE 00000467
-312	Too many tape retries
-313	No statistics available for this device
-314	No statistics available for this unit
-315	Edit program space exceeded
-316	Edit program too large
-317	Edit program version incompatibility
-318	EDITREAD status error
-319	Illegal EXEC string parameter
-320	Bad internal EXEC IPC
-321	Device code is undefined
-322	No matching ?TLOCK (task was not protected)
-323	Not enough disks of this type selected at VSGEN
-324	Task has an outstanding ?UIDCALL
-325	Task has not been called by ?UIDCALL
-326	File inaccessible, run FIXUP on this LDU
-327	LDU; must have FIXUP run on it
-328	Bad transfer count from controller
-329	Invalid baud rate for this device
-330	Illegal operation for this medium
-331	Device not supported
-332	Not a bitmap device
-333	Controller microcode needs to be updated
-334	Illegal label format
-335	Bad diskette
-336	Operator mode already on
-337	Task is not an operator
-338	No matching operator request
-339	Reserved value not zero
-340	Packet version not supported
-341	Cannot rename the root
-342	Cannot delete the root

(continued)

Table A-6 AOS/VS System Error Messages

Error Code	Error Message
-343	Illegal OPEN option for this disk
-344	Modified sector I/O not supported by this device
-345	Modified sector I/O not specified in OPEN
-346	Modified sector I/O requires physical I/O
-347	Invalid number of blocks for modified sector I/O
-348	Tape mounted not scratch tape
-349	Disk released, FIXUP recommended to reclaim space
-350	Not all space freed, FIXUP recommended
-351	Output not held by XOFF - no data to output
-352	Output not held by XOFF - output held by CTS
-353	Output not held by XOFF
-354	Illegal function for device
-355	Error in profile field descriptor packet
-356	Destination field too small (READ)
-357	Source field too large (UPDATE)
-358	Profile access not initiated
-359	Profile access already initiated
-360	Profile entry does not exist
-361	Device is reserved by another port
-362	Illegal open for pipe type file
-363	Illegal pipe size
-364	Pipe is full
-365	Too many requests on pipe
-366	C-type process attempted to create an A-type process
-367	Target program cannot be ringloaded by B- or C-type process
-368	Target program cannot be chained to by B- or C-type process
-369	Attempt to connect A- and C-type processes
-370	Lock is already locked
-371	Lock is not locked
-372	Too many locks for this process
-373	Lock request aborted before granting
-374	Invalid packet ID
-375	Window does not exist
-376	LDU does not exist
-377	Mirrored LDU is not synchronized
-378	LDU name mismatch - not a valid LDU mirror
-379	LDU size mismatch - not a valid LDU mirror
-380	LDU IDs are not unique - not a valid LDU mirror

(continued)

Table A-6 AOS/VS System Error Messages

Error Code	Error Message
-381	Mirrored LDU is out of phase
-382	Mirrored LDU synchronization failed
-383	LDU is not mirrored
-384	Incomplete mirrored LDU specified
-385	LDU format mismatch - not a valid LDU mirror
-386	Controller does not support LDU mirroring
-387	Controller cannot support an additional LDU mirror
-388	Cannot init an LDU image that was previously being synchronized
-389	LDU was released during synchronization - LDU is not synchronized
-390	UNKNOWN MESSAGE CODE 00000606
-391	UNKNOWN MESSAGE CODE 00000607
-392	UNKNOWN MESSAGE CODE 00000610
-393	UNKNOWN MESSAGE CODE 00000611
-394	UNKNOWN MESSAGE CODE 00000612
-395	UNKNOWN MESSAGE CODE 00000613
-396	UNKNOWN MESSAGE CODE 00000614
-397	UNKNOWN MESSAGE CODE 00000615
-398	UNKNOWN MESSAGE CODE 00000616
-399	UNKNOWN MESSAGE CODE 00000617
-400	UNKNOWN MESSAGE CODE 00000620

(concluded)

Table A-7 386/ix System Error Messages

Error Code	Error Message
-1	Not owner
-2	No such file or directory
-3	No such process
-4	Interrupted system call
-5	I/O error
-6	No such device or address
-7	Arg list too long
-8	Exec format error
-9	Bad file number
-10	No child processes
-11	No more processes
-12	Not enough space
-13	Permission denied
-14	Bad address
-15	Block device required
-16	Device busy
-17	File exists
-18	Cross-device link
-19	No such device
-20	Not a directory
-21	Is a directory
-22	Invalid argument
-23	File table overflow
-24	Too many open files
-25	Not a typewriter
-26	Text file busy
-27	File too large
-28	No space left on device
-29	Illegal seek
-30	Read-only file system
-31	Too many links
-32	Broken pipe
-33	Argument out of domain
-34	Result too large
-35	No message of desired type
-37	Channel number out of range
-38	Level 2 not synchronized
-39	Level 3 halted

(continued)

Table A-7 386/ix System Error Messages

Error Code	Error Message
-40	Level 3 reset
-41	Link number out of range
-42	Protocol driver not attached
-43	No CSI structure available
-44	Level 2 halted
-45	Deadlock situation detected/avoided
-46	No record locks available
-50	Bad exchange descriptor
-51	Bad request descriptor
-52	Message tables full
-53	Anode table overflow
-54	Bad request code
-55	Invalid slot
-56	File locking deadlock
-57	Bad font file format
-60	Not a stream device
-61	No data available
-62	Timer expired
-63	Out of stream resources
-64	Machine is not on the network
-65	Package not installed
-67	Link has been severed
-68	Advertise error
-69	Srmount error
-70	Communication error on send
-71	Protocol error
-74	Multihop attempted
-75	Error 75
-76	Error 76
-77	Not a data message
-80	Name not unique on network
-81	File descriptor in bad state
-82	Remote address changed
-83	Can not access a needed shared library
-84	Accessing a corrupted shared library
-85	.lib section in a.out corrupted.
-86	Attempting to link in more shared libraries than system limit
-87	Can not exec a shared library directly

(concluded)

Table A-8 DG/UX System Error Messages

Error Code	Error Message
-1	Not owner
-2	No such file or directory
-3	No such process
-4	Interrupted system call
-5	I/O error
-6	No such device or address
-7	Arg list too long
-8	Exec format error
-9	Bad file number
-10	No child processes
-11	No more processes
-12	Not enough space
-13	Permission denied
-14	Bad address
-15	Block device required
-16	Device busy
-17	File exists
-18	Cross-device link
-19	No such device
-20	Not a directory
-21	Is a directory
-22	Invalid argument
-23	File table overflow
-24	Too many open files
-25	Not a typewriter
-26	Text file busy
-27	File too large
-28	No space left on device
-29	Illegal seek
-30	Read-only file system
-31	Too many links
-32	Broken pipe
-33	Argument out of domain
-34	Result too large
-35	No message of desired type
-36	Identifier removed
-37	Channel number out of range
-38	Level 2 not synchronized

(continued)

Table A-8 DG/UX System Error Messages

Error Code	Error Message
-39	Level 3 halted
-40	Level 3 reset
-41	Link number out of range
-42	Protocol driver not attached
-43	No CSI structure available
-44	Level 2 halted
-45	Deadlock situation detected/avoided
-46	No record locks available
-50	Bad exchange descriptor
-51	Bad request descriptor
-52	Message tables full
-54	Bad request code
-55	Invalid slot
-56	File locking deadlock
-57	Bad font file format
-60	Not a stream device
-61	No data available
-62	Timer expired
-63	Out of stream resources
-64	Machine is not on the network
-65	Package not installed
-66	Object is remote
-67	Link has been severed
-68	Advertise error
-70	Communication error on send
-71	Protocol error
-74	Multihop attempted
-77	Not a data message
-80	Name not unique on network
-81	File descriptor in bad state
-82	Remote address changed
-83	Can not access a needed shared library
-84	Accessing a corrupted shared library
-85	.lib section in a.out corrupted
-86	Attempt to link in more shared libraries than system limit
-87	Can not exec a shared library directly
-90	Too many levels of symbolic links

(concluded)

End of Appendix

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

TIPS ORDERING PROCEDURES

TO ORDER

1. An order can be placed with the TIPS group in two ways:
   a) MAIL ORDER – Use the order form on the opposite page and fill in all requested information. Be sure to include shipping charges and local sales tax. If applicable, write in your tax exempt number in the space provided on the order form.

      Send your order form with payment to:
      Data General Corporation
      ATTN: Educational Services/TIPS G155
      4400 Computer Drive
      Westboro, MA 01581-9973

   b) TELEPHONE – Call TIPS at (508) 870-1600 for all orders that will be charged by credit card or paid for by purchase orders over $50.00. Operators are available from 8:30 AM to 5:00 PM EST.

METHOD OF PAYMENT

2. As a customer, you have several payment options:
   a) Purchase Order – Minimum of $50. If ordering by mail, a hard copy of the purchase order must accompany order.
   b) Check or Money Order – Make payable to Data General Corporation.
   c) Credit Card – A minimum order of $20 is required for Mastercard or Visa orders.

SHIPPING

3. To determine the charge for UPS shipping and handling, check the total quantity of units in your order and refer to the following chart:

Total Quantity	Shipping & Handling Charge
1–4 Units	$5.00
5–10 Units	$8.00
11–40 Units	$10.00
41–200 Units	$30.00
Over 200 Units	$100.00

   If overnight or second day shipment is desired, this information should be indicated on the order form. A separate charge will be determined at time of shipment and added to your bill.

VOLUME DISCOUNTS

4. The TIPS discount schedule is based upon the total value of the order.

Order Amount	Discount
$1–$149.99	0%
$150–$499.99	10%
Over $500	20%

TERMS AND CONDITIONS

5. Read the TIPS terms and conditions on the reverse side of the order form carefully. These must be adhered to at all times.

DELIVERY

6. Allow at least two weeks for delivery.

RETURNS

7. Items ordered through the TIPS catalog may not be returned for credit.
8. Order discrepancies must be reported within 15 days of shipment date. Contact your TIPS Administrator at (508) 870-1600 to notify the TIPS department of any problems.

INTERNATIONAL ORDERS

9. Customers outside of the United States must obtain documentation from their local Data General Subsidiary or Representative. Any TIPS orders received by Data General U.S. Headquarters will be forwarded to the appropriate DG Subsidiary or Representative for processing.

TIPS ORDER FORM

Mail To: Data General Corporation
Attn: Educational Services/TIPS G155
4400 Computer Drive
Westboro, MA 01581 - 9973

BILL TO:	SHIP TO: (No P.O. Boxes - Complete Only If Different Address)
COMPANY NAME _____	COMPANY NAME _____
ATTN: _____	ATTN: _____
ADDRESS _____	ADDRESS (NO PO BOXES) _____
CITY _____	CITY _____
STATE _____ ZIP _____	STATE _____ ZIP _____

Priority Code _____ (See label on back of catalog)

_____ _____ _____ _____ _____
Authorized Signature of Buyer (Agrees to terms & conditions on reverse side) | Title | Date | Phone (Area Code) | Ext.

ORDER #	QTY	DESCRIPTION	UNIT PRICE	TOTAL PRICE

A SHIPPING & HANDLING

☐ UPS	ADD
1-4 Items	$ 5.00
5-10 Items	$ 8.00
11-40 Items	$ 10.00
41-200 Items	$ 30.00
200+ Items	$100.00

Check for faster delivery

Additional charge to be determined at time of shipment and added to your bill.
☐ UPS Blue Label (2 day shipping)
☐ Red Label (overnight shipping)

B VOLUME DISCOUNTS

Order Amount	Save
$0 - $149.99	0%
$150 - $499.99	10%
Over $500.00	20%

Tax Exempt # or Sales Tax (if applicable) _____

ORDER TOTAL	
Less Discount See B	-
SUB TOTAL	
Your local* sales tax	+
Shipping and handling - See A	+
TOTAL - See C	

C PAYMENT METHOD

☐ Purchase Order Attached ($50 minimum)
P.O. number is _____. (Include hardcopy P.O.)
☐ Check or Money Order Enclosed
☐ Visa ☐ MasterCard ($20 minimum on credit cards)

Account Number _____ Expiration Date _____

_____
Authorized Signature
(Credit card orders without signature and expiration date cannot be processed.)

THANK YOU FOR YOUR ORDER

PRICES SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.
PLEASE ALLOW 2 WEEKS FOR DELIVERY.
NO REFUNDS NO RETURNS.

* Data General is required by law to collect applicable sales or use tax on all purchases shipped to states where DG maintains a place of business, which covers all 50 states. Please include your local taxes when determining the total value of your order. If you are uncertain about the correct tax amount, please call 508-870-1600.

Form 702
Rev. 8/87

DATA GENERAL CORPORATION
TECHNICAL INFORMATION AND PUBLICATIONS SERVICE
TERMS AND CONDITIONS

Data General Corporation ("DGC") provides its Technical Information and Publications Service (TIPS) solely in accordance with the following terms and conditions and more specifically to the Customer signing the Educational Services TIPS Order Form. These terms and conditions apply to all orders, telephone, telex, or mail. By accepting these products the Customer accepts and agrees to be bound by these terms and conditions.

1. CUSTOMER CERTIFICATION
Customer hereby certifies that it is the owner or lessee of the DGC equipment and/or licensee/sub-licensee of the software which is the subject matter of the publication(s) ordered hereunder.

2. TAXES
Customer shall be responsible for all taxes, including taxes paid or payable by DGC for products or services supplied under this Agreement, exclusive of taxes based on DGC's net income, unless Customer provides written proof of exemption.

3. DATA AND PROPRIETARY RIGHTS
Portions of the publications and materials supplied under this Agreement are proprietary and will be so marked. Customer shall abide by such markings. DGC retains for itself exclusively all proprietary rights (including manufacturing rights) in and to all designs, engineering details and other data pertaining to the products described in such publication. Licensed software materials are provided pursuant to the terms and conditions of the Program License Agreement (PLA) between the Customer and DGC and such PLA is made a part of and incorporated into this Agreement by reference. A copyright notice on any data by itself does not constitute or evidence a publication or public disclosure.

4. LIMITED MEDIA WARRANTY
DGC warrants the CLI Macros media, provided by DGC to the Customer under this Agreement, against physical defects for a period of ninety (90) days from the date of shipment by DGC. DGC will replace defective media at no charge to you, provided it is returned postage prepaid to DGC within the ninety (90) day warranty period. This shall be your exclusive remedy and DGC's sole obligation and liability for defective media. This limited media warranty does not apply if the media has been damaged by accident, abuse or misuse.

5. DISCLAIMER OF WARRANTY
EXCEPT FOR THE LIMITED MEDIA WARRANTY NOTED ABOVE, DGC MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE ON ANY OF THE PUBLICATIONS, CLI MACROS OR MATERIALS SUPPLIED HEREUNDER.

6. LIMITATION OF LIABILITY
A. CUSTOMER AGREES THAT DGC'S LIABILITY, IF ANY, FOR DAMAGES, INCLUDING BUT NOT LIMITED TO LIABILITY ARISING OUT OF CONTRACT, NEGLIGENCE, STRICT LIABILITY IN TORT OR WARRANTY SHALL NOT EXCEED THE CHARGES PAID BY CUSTOMER FOR THE PARTICULAR PUBLICATION OR CLI MACRO INVOLVED. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO CLAIMS FOR PERSONAL INJURY CAUSED SOLELY BY DGC'S NEGLIGENCE. OTHER THAN THE CHARGES REFERENCED HEREIN, IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING BUT NOT LIMITED TO LOST PROFITS AND DAMAGES RESULTING FROM LOSS OF USE, OR LOST DATA, OR DELIVERY DELAYS, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY THEREOF; OR FOR ANY CLAIM BY ANY THIRD PARTY.

B. ANY ACTION AGAINST DGC MUST BE COMMENCED WITHIN ONE (1) YEAR AFTER THE CAUSE OF ACTION ACCRUES.

7. GENERAL
A valid contract binding upon DGC will come into being only at the time of DGC's acceptance of the referenced Educational Services Order Form. Such contract is governed by the laws of the Commonwealth of Massachusetts, excluding its conflict of law rules. Such contract is not assignable. These terms and conditions constitute the entire agreement between the parties with respect to the subject matter hereof and supersedes all prior oral or written communications, agreements and understandings. These terms and conditions shall prevail notwithstanding any different, conflicting or additional terms and conditions which may appear on any order submitted by Customer. DGC hereby rejects all such different, conflicting, or additional terms.

8. IMPORTANT NOTICE REGARDING AOS/VS INTERNALS SERIES (ORDER #1865 & #1875)
Customer understands that information and material presented in the AOS/VS Internals Series documents may be specific to a particular revision of the product. Consequently user programs or systems based on this information and material may be revision-locked and may not function properly with prior or future revisions of the product. Therefore, Data General makes no representations as to the utility of this information and material beyond the current revision level which is the subject of the manual. Any use thereof by you or your company is at your own risk. Data General disclaims any liability arising from any such use and I and my company (Customer) hold Data General completely harmless therefrom.

134-755-02

moisten & seal

CUSTOMER DOCUMENTATION COMMENT FORM

Your Name _____ Your Title _____

Company _____ Phone _____

Street _____

City _____ State _____ Zip _____

We wrote this book for you, and we made certain assumptions about who you are and how you would use it. Your comments will help us correct our assumptions and improve the manual. Please take a few minutes to respond. Thank you.

Manual Title _____ Manual No. _____

Who are you?
☐EDP/MIS Manager ☐Analyst/Programmer ☐Other _____
☐Senior Systems Analyst ☐Operator _____
☐Engineer ☐End User

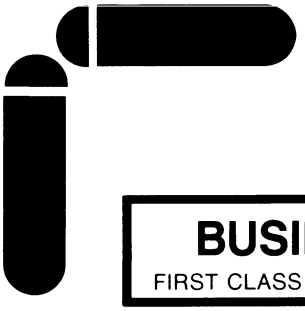
How do you use this manual? *(List in order: 1 = Primary Use)*

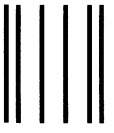
___ Introduction to the product ___ Tutorial Text ___ Other
___ Reference ___ Operating Guide _____

		Yes	No
About the manual:	Is it easy to read?	☐	☐
	Is it easy to understand?	☐	☐
	Are the topics logically organized?	☐	☐
	Is the technical information accurate?	☐	☐
	Can you easily find what you want?	☐	☐
	Does it tell you everything you need to know?	☐	☐
	Do the illustrations help you?	☐	☐

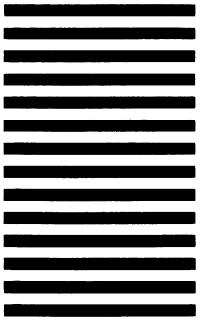
If you wish to order manuals, use the enclosed TIPS Order Form (USA only) or contact your sales representative or dealer.

Comments:





NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 26 WESTBORO, MA 01581

POSTAGE WILL BE PAID BY ADDRESSEE

Data General

Customer Documentation
MS E-111
4400 Computer Drive
P.O. Box 4400
Westboro, MA 01581-9890

Cut here and insert in binder spine pocket

Data General

Data General Corporation, Westboro, Massachusetts 01580

093-000685-00