IRIS R8 *User Manual*

IRIS R8
USER MANUAL

Revision 04

NOTICE

Every attempt has been made to make this manual complete, accurate and up-to-date. However, all information herein is subject to change due to updates. All inquiries concerning this manual should be directed to POINT 4 Data Corporation.

POINT 4 Data Corporation
15442 Del Amo Avenue
Tustin, CA 92680
(714) 838-2225

REVISION RECORD

PUBLICATION NUMBER: SM-030-0011

Revision	Description	Date
01	First Draft Version	07/13/82
02	Preliminary Release (withdrawn)	03/01/83
03	Complete Revision	05/10/85
04	Update Package to incorporate the index	08/10/85

LIST OF EFFECTIVE PAGES

Changes, additions, and deletions to information in this manual are indicated by vertical bars in the margins or by a dot near the page number if the entire page is affected. A vertical bar by the page number indicates pagination rather than content has changed. The effective revision for each page is shown below.

Page	Rev	Page	Rev	Page	Rev
Cover	-				
Title	04				
ii	03				
iii,iv	04				
v thru ix	03				
x	04				
xi	03				
1-1 thru 1-11	03				
2-1 thru 2-36	03				
2-37	04				
2-38 thru 2-123	03				
3-1 thru 3-23	03				
Appendix Title	-				
A-1 thru A-8	03				
B-1 thru B-5	03				
C-1 thru C-8	03				
D-1	03				
Index-1 thru Index-5	04				
Comment Sheet	04				
Mailer	-				
Back Cover	-				

PREFACE

The IRIS User Manual is intended for all IRIS users. Included are sections on IRIS conventions and procedures, system commands, and editors that are standard with an IRIS system.

Section 2, IRIS System Commands, contains general purpose commands such as LIBR and SAVE. It also describes commands that are used for special purposes, such as ANALYPF and XREF. The section is arranged in alphabetical order for easy reference.

The appendices contain an IRIS glossary, terminal key commands, and ASCII codes under IRIS.

For specific instructions on the installation and configuration of an IRIS system, please refer to the IRIS R8 Installation and Configuration Manual. Daily operations such IPL, backups, and the general maintenance of the system are discussed in the IRIS Operations Manual. Optional application packages offered by POINT 4 are described in separate manuals.

The term BASIC as used in this manual refers to IRIS Business BASIC.

Standard Notations for This Manual

This manual uses the following standard writing conventions:

User Input	User input is always underlined. It may be a command shown in capital letters, a variable such as a filename shown in braces, or locations in memory indicated by an octal number.
<RETURN>	Indicates a carriage return. It is required to activate command input. This is not shown unless it is the only command required, a second <RETURN> is required, or it follows a control character (i.e., <CTRL-Z> <RETURN>).
<CTRL-x>	Indicates a control character where x is an alpha key. It is entered by holding down the CTRL key and pressing the alpha key indicated. Both keys are then released. A <RETURN> is not required unless otherwise noted.
variable	Lowercase string represents a variable such as a filename, password, etc.
{option}	Lowercase string enclosed in braces represents an optional parameter.

Related Manuals

Related manuals include:

Title	Pub. Number
IRIS Installation and Configuration Manual	SM-030-0009
IRIS Operations Manual	SM-030-0010
IRIS Business BASIC Manual	SM-030-0012
POINT 4 Computer User Manual	HM-080-0003
MARK 8 Computer Reference Manual	HM-082-0021
POINT 4 Supplement to SMC BASIC Manual	AM-190-0027
Thoroughbred SMC BASIC Reference Manual	

CONTENTS

APPENDICES

INDEX

FIGURES

TABLES

Section 1
INTRODUCTION AND IRIS CONVENTIONS

1.1 INTRODUCTION

IRIS is an acronym for the Interactive Real-Time Information System. IRIS is a high-performance system that supports an interactive, multiuser, time-sharing environment.

IRIS supports two BASIC language interpreters and one preprocessor. The two interpreters are for IRIS Business BASIC and SMbasic; the ABASIC preprocessor is for IRIS Business BASIC. In this document, "BASIC" refers to IRIS Business BASIC. SMbasic and ABASIC are referred to by their individual names.

IRIS also supports a variety of file types, which allows flexible and fast access to data. IRIS file types include text, formatted, indexed, contiguous, and polyfiles. Refer to the IRIS Business BASIC Manual for a detailed description of IRIS files.

The IRIS Operating System is fast and efficient. The guidelines provided in this and other IRIS manuals allow the user to take full advantage of the system's many features. A glossary of IRIS terms is provided in Appendix A.

Under IRIS, a number of control keys have certain predefined functions. Appendix B describes the commands and their functions.

If the system detects an error condition while a system command or IRIS Business BASIC statement is being executed, a numeric error code is generated. The IRIS Business BASIC Manual contains a list and a description of these error codes. The Thoroughbred* SMC BASIC Reference Manual and POINT 4's Supplement to the SMC BASIC Manual contain information about SMbasic error codes. System halts and traps are described in the IRIS R8 Operations Manual.

*Thoroughbred is a Trademark of Science Management Corporation.

1.2 USER ACCOUNTS

Each user is assigned an account by the system manager. Several users may be assigned to the same account and use the system simultaneously from different terminals. Account information is contained in a file (ACCOUNTS) that resides on each logical unit. A logical unit is a partition of a physical disk. Some of this information may be displayed when the user logs on or off the system, as described in Sections 1.5 and 1.6. The information contained in the ACCOUNTS file includes:

- Account ID - access code consisting of an alphanumeric string of up to 12 characters that is assigned to a user by the system manager. The account ID string may consist of upper and lowercase characters. It must be entered exactly as defined by the system manager.

- Privilege level - a numeric code from 0-3 that dictates what files may be accessed.

  0 - may access own files and those not protected against such use.

  1 - may access all level 0 files of the same group and those not protected against such use.

  2 - may examine and modify any level 0 or 1 files; has limited access to certain system files.

  3 - may access all files (restricted to system manager).

- Account number - identifies files created by a user.

- Connect and CPU time - each user account is allotted from 0 to 1000 (or unlimited) hours of connect and CPU time by the system manager.

  A user's account is charged for connect time from log-on until log-off, regardless of whether the system is used during that time. CPU time is charged only when the computer is actually performing a task for the user. Allotted CPU or connect time may expire while a user is logged on. The user is allowed to continue until log-off. This overtime condition is then displayed. No user will be allowed to log on to that account until the system manager revises the time limitation.

- Assigned priority - a priority from 1-7 is assigned to each account by the system manager. It is used by the system scheduler to calculate priority level when the user's program is enqueued for CPU time. Programs run from high priority accounts generally complete sooner.

- Disk blocks allotted and remaining - each account is allotted a number of disk blocks by the system manager for saving programs and building data files on one or more logical units. The number of disk blocks in use on a user's account starts at zero and is updated by the system each time a file is built, extended, or deleted. The allotted number of blocks cannot be exceeded. Total allotment for all accounts may exceed the amount of physical disk space.

- Total file use charge - each time a user opens a file belonging to another account, the cost of that file is added to net charges in the user's own account. The income earned by a file is accrued in the file's header block. This information may be used in a service bureau environment for billing purposes and by the system manager for analysis.

- Assigned logical unit - a user is assigned to a logical unit (LU) by the system manager when the user's account is created. This is the LU that is accessed automatically when the user logs on. The user may be allotted a number of disk blocks for building files on this LU.

  Each user must have an account on logical unit zero in order to log on to the system but should not have disk space allotted on it.

  Refer to the glossary in Appendix A for a definition of a logical unit.

1.3 USER TERMINALS UNDER IRIS

A user's terminal under IRIS may be in one of the following three basic states:

1. Idle state - the terminal is in an idle state when it is not logged on. The computer responds only when <ESC> is pressed. A welcome message is displayed, followed by a prompt for an account ID.

2. Command mode - after the account ID is entered, the user is logged on and the terminal is in command mode, as indicated by the system command prompt (#). Only in this mode will system commands, such as BASIC, SAVE, KILL, LIBR, BYE, etc., be accepted.

   Most processors have been given a verb as a filename (e.g., COPY, QUERY) to make their function obvious. In this manual, the command that activates a processor is shown in uppercase letters, although the command itself may be typed on the keyboard in lowercase letters unless otherwise noted. To activate (i.e., enter) a system command or a response to a prompt, a <RETURN> must be pressed unless otherwise noted.

   In some cases, the user may return to command mode by pressing <CTRL-C> (hold down the control key and press the C key).

3. Processor mode - most system or language commands invoke a machine code program called a processor or an IRIS Business BASIC program. Some processors, such as SAVE and KILL, perform their entire task and return to command mode. Others are utility programs that are interactive and prompt for certain parameters. Upon completion, control returns to command mode.

1.4 IRIS FILENAMES

A device or disk file is accessible by means of its unique name. This name, and an optional password, must conform to IRIS conventions. It is assigned by a user when the file is first created.

A filename is a string that may consist of alpha characters, numerics, and periods to a maximum length of 14 characters (13 characters for a polyfile). The first character of a filename must be alpha with one exception: peripheral drivers or system subroutines that are currently enabled require that their names start with a dollar sign ($) followed by an alpha character. If a password is used, the maximum 14-character length includes the filename, password, and <CTRL-E> codes. A <CTRL-E> immediately preceding the password protects and prevents it from being displayed. Entering a second <CTRL-E> following the password reenables the echo and allows subsequent entries to be displayed.

Examples of acceptable filenames are:

```
R8.ACCTS.1
XY14.99
AB.123c
RESEARCH<CTRL-E>YY<CTRL-E>
$LPT
```

With the exception of polyfiles (see Section 1.4.1), the same filename may be used on different logical units for different files.

To access a file or a program, at the system command prompt (#), enter

filename

When the filename is entered in this format, the system searches logical unit zero (LU 0) first, then the default logical unit, and finally the user's assigned logical unit (LU).

If it is necessary to use a file (or create a new one) on a logical unit other than the user's assigned LU, enter the filename in the format

lu/filename

where

lu - number of the logical unit on which the file is to be found or built

Possible logical unit numbers range from 0 to 127. However, LU 0 is generally reserved for system files.

1.4.1 POLYFILE NAMES

Polyfile names are subject to the same conventions as other IRIS files. A polyfile name must end with an @. A two-digit volume number is added by the system. The maximum number of characters allowed is 13, excluding the @. If a password is used, the 13 characters before the @ must include the password and the <CTRL-E> codes.

Examples of acceptable polyfile names are:

```
PF.APPLE@
PF.ABC.6@
PF.ACCT<CTRL-E>XY<CTRL-E>@
```

An exclamation mark (!) at the end of a polyfile name is used when the file is opened for read-only access. The use of an ! with a polyfile name does not overlay an existing file.

When a polyfile is built using the BUILDPF command (see Section 2.6), the format for its name is

```
lu/name@
```

where

lu - number of the logical unit on which the polyfile is to be created.

Possible LU numbers range from 1 to 127.

The name of a polyfile must be unique on those LUs on which the polyfile volumes are to be built. If a file with the name ABC exists on LU 3 and a volume of your polyfile is to be built on that logical unit, you may not call the polyfile ABC@. The @ does not make the filename unique and the name ABC@ will be rejected by the system. Similarly, if a polyfile named ABC@ exists on a logical unit, another file (e.g., text, contiguous, formatted) named ABC cannot be built.

1.4.2 FILE COST AND PROTECTION

It may be necessary to give a file a protection status and, for accounting purposes, indicate the amount to be charged each time that file is accessed.

Cost and protection are optional. The default for cost is zero. The protection level codes are combined into a two-digit number. The first digit represents the protection against lower privilege users. The second specifies protection against users with the same privilege level as the originator of the file (e.g., 77 gives maximum protection and is the system default).

The procedure for specifying cost and protection when building a formatted or contiguous file is described under the FORMAT command.

Cost and protection may be added to any file by using the CHANGE command.

TABLE 1-1. PROTECTION LEVEL CODES

Code	Description
0	No protection
1	Copy-protect
2	Write-protect
4	Read-protect
3	Copy- and write-protect
5	Read- and copy-protect
6	Read- and write-protect
7	Read-, write-, and copy-protect

1.4.3 OVERWRITING FILES

When a file is being built or copied, an exclamation mark (!) following the filename (other than a polyfile) allows the file to replace an existing file of the same name and type on the user's own account.

1.5 IRIS LOG-ON PROCEDURE

The IRIS log-on procedure is as follows:

1. Turn the terminal's power switch to the On position.

2. Press <ESC> or <ALT MODE>.

   On a CRT or terminal without an escape key, try one of the following alternatives:

   - Press <CTRL-D> (on any terminal)
   - Press <CTRL-[> (some terminals)
   - Press <CTRL-SHIFT-K> (some terminals)

   A control character is entered by holding down the <CTRL> (control) key while pressing a given character and then releasing both. The CTRL-SHIFT action is similar, but both the CTRL and the SHIFT keys must be held down while the given character is pressed and then all three keys may be released. (Appendix B provides a description of all terminal command keys.)

   The system displays an optional welcome message, and then prompts

   ```
   ACCOUNT ID?
   ```

3. Enter the appropriate account ID and press <RETURN>. The account ID is not displayed. It must be entered exactly as specified by the system manager. Lowercase and uppercase characters are not interchangeable. An invalid or misspelled account ID causes INVALID to be displayed. The ACCOUNT ID? prompt is then repeated.

   Upon entry of a valid account ID, account information may be displayed as shown in Figure 1-1. (The system manager has the option of suppressing some or all of this account information.)

4. The system command prompt (#) indicates that log-on has been completed and the terminal is in command mode.

```
ACCOUNT ID?  PORT #nn       GROUP n  USER nn
mon dd, yyyy  hh:mm:ss

CPU TIME AVAILABLE      - nnnnnn
CONNECT TIME AVAILABLE  - nnnnnn

nnnnnn BLOCKS IN USE, nnnnn AVAILABLE ON UNIT #n

#
```

Figure 1-1. Log-on Display

A log-on problem occurs when the computer does not accept an account ID or echoes a password. It may be possible to correct the problem by one of the following methods:

- If the system bleeps or does not respond, it may not have recognized the ESC key. First try pressing <ESC> again. If the bleep repeats, the cause may be incorrect parity checks.

  To toggle the parity check, press <CTRL-P>.

  A percent sign (%) is then echoed. Press <ESC>, enter your account ID, and press <RETURN>. The log-on process should now proceed normally. If <CTRL-P> is pressed a second time, it cancels the first <CTRL-P>.

- If the computer still does not respond, the cause might be:

  - The time-sharing system may be shut down for use in stand-alone mode or for maintenance.

  - The terminal may not be connected properly to the computer.

  If the computer is in stand-alone mode (e.g., backups are being performed), wait until operations return to normal.

  If the computer responds in a manner other than outlined above, it is possible that another user left the terminal without logging off. Press <CTRL-C> twice and enter BYE to log off. Then, log on with the proper account ID.

- The terminal's baud-rate switch may have been set incorrectly. Some CRT terminals have a switch labeled "baud rate".

  Press either <ESC> or <CTRL-P> and <ESC> twice with the switch at each setting. If your system has a POINT 4 MIGHTY MUX and autofrequency is enabled, the <BREAK> key may be used to try the next baud rate. If a modem is used on the system, press <BREAK><ESC> until the ACCOUNT ID prompt is displayed. The PORT command may also be used for changing the baud rate.

- If your account ID is displayed as you type it in, press <CTRL-E><ESC> and reenter the account ID. If this does not solve the problem, the terminal may be in half-duplex mode. Look for a switch labeled HD-FD and set it to the FD (full-duplex) position.

- If the system responds with a beep after the account ID is entered and displays

  ```
  ACCOUNT ID?  INVALID
  ACCOUNT ID?
  ```

  reenter your correct account ID. Make sure it is entered as specified by the system manager (i.e., uppercase, lowercase, or both).

- If your assigned logical unit (LU) has not been installed, the system prints the message

  ```
  LOGICAL UNIT NOT ACTIVE
  PRESS RETURN TO LOG ON; ELSE PRESS ESC
  ```

  Notify the system manager that your LU is not installed. After installation, repeat the log-on process.

- If your account has been restricted to the use of certain ports or to a particular time of day, it will be impossible to log on to other ports or at other times. One of the following error messages will be displayed:

  ```
  ACCOUNT IS INVALID ON THIS PORT
  ```

  or

  ```
  ACCOUNT MAY ONLY BE USED FROM hh:mm TO hh:mm

  PORT MAY ONLY BE USED FROM hh:mm TO hh:mm
  ```

  where
  hh:mm - time (hours and minutes)

  Ask your system manager whether your account is restricted.

1.6 LOG-OFF PROCEDURE

It is necessary to log off before leaving the terminal to avoid being charged for additional connect time, to prevent unauthorized personnel from using your account, to prevent access to your files, etc.

The BYE command is used to log off. At the system command prompt (#), enter

BYE

Generally, the account status is then displayed as shown in Figure 1-2. The system manager has the option to suppress portions of the accounting information.

NOTE

If the 'Auto-Log-Off' option is set in the Port Control Word, modems are logged off when disconnected.

```
#BYE  GROUP n  USER nn          mon dd, yyyy  hh:mm:ss

NET ACCRUED CHARGES:   $$$.cc

CPU TIME USED        h:mm:ss
CONNECT TIME USED    h:mm:ss

nnnnnn BLOCKS IN USE, nnnnnn AVAILABLE ON UNIT #n
```

Figure 1-2. Log-off Display

Section 2
IRIS SYSTEM COMMANDS

The commands discussed in this section are available to all users of the IRIS system. Some commands are general purpose commands, such as LIBR; others have specific applications, such as ANALYPF. The commands are arranged in alphabetical order for easy reference.

An IRIS system command is a string that constitutes the filename of a processor or utility program. It is entered at the system command prompt (#) and followed by <RETURN>. If entered correctly, it will activate the selected processor or program. However, the command may be rejected for any of the following reasons:

- Command is not the name of a program or processor.
- Command was entered incorrectly.
- Processor is not on logical unit zero (LU 0), not on the user's assigned logical unit (LU), not on the default LU, and not on the specified LU.
- Program or processor is protected or requires a password.

If the command is rejected, IRIS displays the message

```
? NO SUCH PROCESSOR
```

The system command prompt (#) is then displayed.

Commands, debuggers, and utility programs used for system configuration procedures are described in the IRIS R8 Installation and Configuration Manual. Command and utility programs required for system maintenance are discussed in the IRIS R8 Operations Manual. Commands that are activated via a language processor, such as BASIC's NEW, SIZE or RENUMBER, are discussed in the appropriate language manuals.

The following sections discuss each system command and its application. Where necessary, a summary of an IRIS file structure is given. For a more comprehensive description of IRIS files, refer to the IRIS Business BASIC Manual.

2.1 ABASIC

ABASIC is a preprocessor for IRIS Business BASIC. The user first creates a source text file by using an IRIS text editor, such as EDIT (see Section 3.1). The source file is then compiled into IRIS Business BASIC by entering the following command and string at the system command prompt (#):

ABASIC {source filename}

If the name of the source file is not included in the command string, ABASIC will prompt for a filename.

Refer to the ABASIC Tech Memo in the R8.2 Release Notes for information on using the ABASIC preprocessor.

2.2 ALOAD

ALOAD is a utility program that may be used to load an assembly language program into an existing file, such as REX or DISCSUBS. ALOAD does not overwrite cells for which the IRIS Assembler has not generated any values (i.e., contents are 77377 (octal)).

Two parameters must be entered by the user. The first is the source file, which must be a stand-alone program or a processor (i.e., a type 3 file). The other parameter is the destination file, which may be any file type.

ALOAD displays addresses as it is loading the requested file. These addresses are typically at block boundaries. ALOAD should be saved on a manager or utility account at protection level 77.

To use ALOAD, at the system command prompt (#), enter

```
ALOAD
```

The program then displays

```
ALOAD - * A FILE LOAD

File to load from [must have "LU/"]:
```

Enter the appropriate source logical unit number followed by the desired filename. If the file is found and it is a type 3 file, ALOAD displays

```
File to load into [must have "LU/"]:
```

Enter the appropriate destination logical unit followed by the desired filename. If the file is found and it is not a contiguous file, ALOAD displays

```
Loading .....
```

At the successful completion of the procedure, the system command prompt (#) is displayed.

If the source file is not found, or is of the wrong type, the prompt "File to load from" is redisplayed. Similarly, if the destination file is not found, or is a contiguous file, the prompt "File to load into" is redisplayed. No specific error messages are given.

To exit the program without entering the parameters, press <CTRL-C>.

2.3 ANALYPF

ANALYPF is a superset of the BASIC program, QUERYPF. When ANALYPF is invoked, QUERYPF options (i.e., global display or complete dump) may be selected. However, ANALYPF has an added function that may be used to analyze the structure of polyfile directory volumes (base and extension). This feature allows the user to examine every key by displaying the contents of the directories.

The QUERYPF options may be used to find the correct volume number of the directory to be analyzed. Section 2.23 provides information on using those functions.

This section describes the Analyze function, which can only be invoked from ANALYPF.

2.3.1 INTRODUCTION TO ANALYPF PROCEDURES

The first directory to be analyzed must be the appropriate base directory because the search for a key or keys must begin with the master level directory and the master level block. In a polyfile, the fine level number is zero; the master level number depends on the number of intermediate levels. Therefore, the master level number is always the highest level number in a given directory. The master level number is <u>not</u> related to a directory volume number. Refer to the Polyfile Document for information on the polyfile directory structure.

ANALYPF displays linkage information for the various directory levels, including volume and block numbers for the forward and backward links, the record number of each key in a specified directory block, and the value of the keys in that block.

If the directories to be analyzed do not contain any keys, no useful information is returned. For example, if a polyfile was structured during a COPYPF procedure but no data has been entered, ANALYPF cannot display linkage information or key values.

The program does not analyze the structure of data volumes. If a volume or block number associated with a data volume or map block is entered at a prompt, ANALYPF displays an appropriate message and repeats the prompt for a volume or block number.

The Analyze function prompts for the following parameters:

- Volume number
- Block number of the master level directory
- Block number for intermediate and/or fine level directory

Each directory level has a block number associated with it. To find the block number associated with the master level, use the information displayed after the volume number has been entered. The information displayed includes:

- Size of the volume in blocks
- Number of blocks containing keys

A base directory volume consists of a header, from 1 to 16 blocks for maps, and 1 or more base directories. Each base directory contains the master level block and the last fine level block. To find the master level block number of the first directory in the volume, two steps are required:

1. Calculate the number of blocks used for the bit maps by using the algorithm

   T - H - K = number of map blocks

   where
   T - volume size in blocks
   H - (header block)
   K - number of blocks used for the keys

2. Calculate the master level block number by adding one to the number of map blocks. For example, if there are five map blocks numbered 0-4, add one to the highest numbered map block (i.e., 4); thus, the master level block number is 5.

To find the master level block number for the other directories in the volume, add two blocks for each subsequent directory (one master and one fine level block) to the first directory's master level block number.

For example, a specified volume has three directories and five map blocks (0-4). The directories are numbered 1, 6, and 8. To examine directory 8, add six blocks to the first directory's master level block. In this example, the master level block of directory 8 is 11 (decimal).

2.3.2 USING ANALYPF

This section shows the use of ANALYPF by giving an example of the program prompts, messages, and user input (underlined). The base directory volume contains two directory levels: master and fine. When the master level directory is displayed, no backward link is shown.

To display the characteristics of an individual volume, enter the volume and block numbers in decimal or octal numbers. If a decimal number is entered, it must be followed by a period. The program also displays the decimal equivalent of the octal number. For example, if 0 is entered at a volume number prompt and 10 at a block number prompt, the program displays

```
Volume 000 (  0.)     Block 000010 (     8.)
```

If 0. is entered at a volume number prompt and 8. at a block number prompt, the program displays

```
Volume 000 (  0.)     Block 000010 (     8.)
```

To use the ANALYPF program, at the system command prompt (#), enter

```
ANALYPF
```

The program then displays the following messages and prompts:

```
ANALYPF - Analyze Polyfile Utility


Polyfile name [should have "LU/"]: 1/TESTANPF@

Output file [<RETURN> = output to terminal]: <RETURN>
Scanning polyfile, please wait...
Please input volume number [0-63]
     or   <RETURN> for global display
     or      -1 for complete dump
     or      -2 for analysis mode

      or ESCape to exit to SCOPE: 0
```

To display the characteristics of a particular volume, press <RETURN> at the "Output file" prompt and enter the desired volume number in decimal. The information may then be used to calculate the master level block number, which directories require analysis, etc.

An example of an individual volume display is as follows:

```
Volume: 0      DHDR:  1/001130          Logical unit   1 installed.
Privilege level: 2      Group: 1  User: 4    Protection: 77
Size: 13 disk blocks
Volume is a Base Directory volume with 2 directories.
Base Volume 0 and its current extensions have a total of 11 blocks
for keys which will hold a maximum of approximately 99 keys.
Directory:  Key length (in characters)
      0: 20      2:20
```

Using the information displayed here, the master level block number of the first directory can be calculated as block 1 (i.e., 11+1 = 12; 13-12 = 1).

The dialog contines as follows:

```
Scanning polyfile, please wait...
Please input volume number [0-63]
     or   <RETURN> for global display
     or       -1 for complete dump
     or       -2 for analysis mode

      or ESCape to exit to SCOPE: -2

TVolume # (octal): 0
TBlock # (octal): 1

Volume 000 (  0.)     Block 000001 (     1.)
Master block Directory 1     Level 1
 5  keys of length 20
Forward link:   Volume 000  Block 000000
Backward link:  Volume 000  Block 000000
 <RETURN> for block contents:  <RETURN>

000  000002    "KEY    10"
000  000005    "KEY    20"
000  000007    "KEY    30"
000  000011    "KEY    40"
000  000013    *** Terminator key ***

TBlock # (octal): 13
```

Note that the volume number, block number, and last key in that block are displayed. To check the last fine level block, enter 13. The program displays the information about the block. The block's contents may then be displayed.

```
Volume 000 (  0.)     Block 000013 (     11.)
Fine block    Directory 1    Level 0
 11 keys of length 20
Forward link:   Volume 000  Block 000000
Backward link:  Volume 000  Block 000011
 <RETURN> for block contents:  <RETURN>
    40            "KEY      41"
    41            "KEY      42"
    42            "KEY      43"
    43            "KEY      44"
    44            "KEY      45"
    45            "KEY      46"
    46            "KEY      47"
    47            "KEY      48"
    48            "KEY      49"
    49            "KEY      50"
    16777215      *** Terminator key ***
    TBlock # (octal): 3
```

Assume that the key values are correct in this directory and the next directory is to be examined. The master level block number of the second directory is 3 (1+2). After the appropriate block number is entered, the dialog continues:

```
Volume 000 (  0.)     Block 000003 (     3.)
Master block  Directory 2   Level 1
 4 keys of length 20
Forward link:   Volume 000 Block 000000
Backward link:  Volume 000 Block 000000
 <RETURN> for block contents:  <RETURN>
    000  000004   "BAKER"
    000  000006   "MARY"
    000  000010   "ROSE"
    000  000012   *** Terminator key ***
```

Enter the number of the block where the problem key may be located as shown in the following example:

```
TBlock # (octal): 10

Volume 000 (  0.)     Block 000010 (     8.)
Fine block    Directory 2   Level 0
 10 keys of length 20
Forward link:   Volume 000  Block 000012
Backward link:  Volume 000  BLock 000006
 <RETURN> for block contents:  <RETURN>
    20            "MAXI"
    21            "MOE"
    22            "NELLY"
    23            "NOH"
    24            "OTTO"
    25            "PAL"
    26            "PETRA"
    27            "PROBBELEM"
    28            "QUIMBY"
    29            "ROSE"
```

The program again displays information about the directory level block. Press <RETURN> to see the contents of the block. This is a fine level directory and the number to the left of the key value is the record number for the key. If the search for the problem key had included an intermediate level, volume and block information would point to the next levels.

Record number 27 (key value PROBBELEM) was misspelled. The error may now be corrected with the appropriate SEARCH mode (see the IRIS Business BASIC Manual).

Press <RETURN> at the block and volume prompts and then press <ESC> to exit the program as shown in the following example:

```
TBlock # (octal): <RETURN>
TVolume # (octal): <RETURN>
Scanning polyfile, please wait...
Please input volume number [0-63]
     or   <RETURN> for global display
     or      -1 for complete dump
     or      -2 for analysis mode

      or ESCape to exit to SCOPE: <ESC>
#
```

If a volume or block number is entered that references a data volume or map block, ANALYPF identifies the error and redisplays the appropriate prompt as shown in the following example:

```
TBlock # (octal): 1

Volume 001 (  1.)     Block 000001 (     1.)
Data volume.

TBlock # (octal): 0

Volume 001 (  1.)     Block 000000 (     0.)
MAP BLOCK
Data volume.

TBlock # (octal): <RETURN>
TVolume # (octal): 0

TBlock # (octal): 0

Volume 000 (  0.)     Block 000000 (     0.)
MAP BLOCK

TBlock # (octal): <RETURN>
TVolume # (octal): <RETURN>
Scanning polyfile, please wait...
Please input volume number [0-63]
     or   <RETURN> for global display
     or      -1 for complete dump
     or      -2 for analysis mode

      or ESCape to exit to SCOPE: <ESC>
```

2.4 ASM

ASM is the IRIS disk-to-disk machine language assembler. It is fully compatible with the Nova* stand-alone absolute assembler.

For a description of ASM instructions, refer to the POINT 4 Computer Reference Manual or the POINT 4 MARK 8 Computer Reference Manual. Information on permissible opcode parameters is contained in the 0/SYMBOLS text file. The file can be printed.

The IRIS editor, EDIT, can be used to create an ASM program in text file format.

To list or assemble an ASM program, at the system command prompt (#), enter

ASM {objectfile}, {@listfile,}{-}source1{,{-}source2}....

where

@ - specifies the following filename to be the destination of the listing

\- (minus sign) - suppresses listing of the source file that it precedes

NOTE

All specified source files are processed sequentially.

*Nova is a trademark of Data General Corporation.

2.5 BASIC

IRIS Business BASIC is one of two BASIC programming languages supported under IRIS. For a detailed description of its use and functions, refer to the IRIS Business BASIC Manual.

To input, list, or modify a BASIC program in text file format, at the system command (#), enter

```
BASIC
LOAD filename
```

To list or modify a saved BASIC program, at the system command prompt (#), enter

```
BASIC filename
LIST
```

2.6 BUILDPF

BUILDPF is a utility program used for the creation or extension of a polyfile. A summary of the polyfile features is provided as an aid to determining the appropriate parameters. A BUILDPF exercise is provided in Appendix C.

2.6.1 SUMMARY OF POLYFILE FEATURES

A polyfile is made up of one or more associated files. Each file is a contiguous file called a volume. The maximum number of volumes is 64 (0-63). Each volume may reside on a different logical unit (1-127). The various volumes are tied together into one entity by the filename, last access date, and pointers listed in a master volume. The master volume is always volume 0.

The various types of volumes are the following:

- Base directory - contains the master and first fine block for one or more directories.

- Directory extension - a directory extension volume provides additional disk blocks that can be used to extend a base directory volume.

- Data volume - the same record size is used for all data volumes in a given polyfile. Records are numbered sequentially through ascending data volume numbers. Therefore, it is recommended that the volume numbers be assigned by default.

  Data volumes can be built with bit maps. The polyfile allocation routine uses the bit maps to list records that are in use and those that are available for allocation. In any one polyfile, all data volumes must be either mapped or unmapped.

Points to remember when building a polyfile include the following:

- A polyfile must have a unique name.

- It is more efficient to use the map option for data volumes. This allows the system to keep track of free records. If the map option is not used, the application program must provide for tracking free records.

- The maximum number of volumes in a polyfile is 64 (volumes 0-63). The master volume is always volume number 0. The size of a volume is limited by the size of the logical unit (LU) on which it resides (maximum size = 65335 blocks, including the header and map blocks).

- A record size must be given for the master volume even if the polyfile will not contain any data volumes.

  The maximum size of data volumes and the master volume (volume 0) is limited by the record length specified. If volume 0 is built as a directory, or a directory extension volume, the number of keys in volume 0 is restricted by this size limitation. The approximate maximum size of a data or master volume is

$$MAXB = \begin{cases} 65535 \text{ x(r)/256} \\ \text{to a maximum of} \\ 65535 \text{ blocks} \end{cases}$$

  where
  MAXB - maximum size in blocks
  r - record size in words

  Thus, a polyfile with a small record size may need more data volumes to use up the same disk space (i.e., number of blocks) as the data volumes of a polyfile with a large record size.

- It is recommended that volume 0 be specified as a data volume and the first base directory be a nonzero volume (i.e., 1-63).

- Keys in the directory volumes (base or extended) may be up to 121 bytes in length.

- It is advisable to let the BUILDPF program assign volume numbers.

- To maximize polyfile performance, keep the number of data and directory extension volumes as small as possible. Fewer data volumes containing a large number of records provide better performance than many volumes containing fewer records. This is especially important when getting free records to add new records. Similarly, one large base directory volume is more efficient than a small base directory with one or more directory extension volumes.

2.6.2 USING BUILDPF

At the system prompt (#), enter

```
BUILDPF
```

The program responds

```
BUILDPF - Build Polyfiles Utility
```

It then requests a filename with the prompt

```
POLYFILENAME [must have "LU/" (not 0)]:
```

A polyfile should not be built on LU 0. The range of possible LUs is from 1 to 127. The name must be terminated by an at sign (@).

BUILDPF then attempts to open the file. If the polyfile is found, BUILDPF enters the polyfile extension mode as described in Section 2.6.6. If the file is not found, BUILDPF prompts

```
POLYFILE NOT FOUND DO YOU WISH TO CREATE A NEW ONE? (Y/N)
```

If the answer is N, the program returns control to system command mode. If the answer is Y, BUILDPF requests a record size

```
RECORD SIZE (in words for the entire polyfile):
```

After the record size has been entered, BUILDPF displays a message indicating the volume being built

```
VOLUME: 0
```

Volume 0, the master volume, is always built first. Next, BUILDPF requests volume type information.

2.6.3 VOLUME TYPE INPUT

BUILDPF prompts for the type of volume to be built

```
VOLUME TYPES:  "B"  Base Directory
               "E"  Extension Directory
               "D"  Data Volume
VOLUME TYPE:
```

2.6.3.1 Base Directory

To build a base directory, enter B. BUILDPF then prompts

```
STARTING DIRECTORY NUMBER FOR THIS VOLUME:
```

Enter the lowest possible directory number. The directory number must be in the range of 1 to 63 and cannot be currently in use.

BUILDPF then displays

```
DIRECTORY NUMBERS AVAILABLE FROM xx THRU yy
```

where
- xx - the specified starting directory number
- yy - the range of contiguously available directory numbers

BUILDPF then prompts for the key size for the volume.

```
DIRECTORY xx KEY SIZE IN CHARACTERS [<RETURN> TO TERMINATE]:
```

Enter any valid key size in the range of 2 to 121.

BUILDPF increments directory numbers automatically by one and prompts for the next directory's key size until <RETURN> is pressed.

```
THIS DIRECTORY SETUP OK?
```

Enter <RETURN>, Y, or y to approve the directory setup. Any other input deletes the parameters entered and the program prompts

```
STARTING DIRECTORY NUMBER FOR THIS VOLUME
```

2.6.3.2 Directory Extension

To build a directory extension volume, enter E. BUILDPF then prompts

```
VOLUME TO EXTEND:
```

The volume to be extended must be an existing base directory volume.

2.6.3.3 Data Volume

To build a data volume, enter D. BUILDPF then prompts

```
DATA VOLUME(S) TO HAVE MAPS? ["Y"/"N"]:
```

This prompt appears only once because the answer given to this question will apply to all data volumes in the polyfile.

2.6.4 VOLUME SIZE INPUT

Input for volume size information depends on the volume type. For data volumes, the size parameter may be entered by stipulating the maximum number of records or the number of blocks (less header) desired. For directories, size requirements may be entered by stipulating the maximum number of keys or number of disk blocks (less header) desired.

2.6.4.1 Base Directory Volume

BUILDPF requests the volume size for a base directory volume

```
VOLUME SIZE IN INDEXES (KEYS) [NEGATIVE FOR SIZE IN BLOCKS]:
```

If the volume size is given in number of keys, the size of the volume is computed by multiplying the key size previously specified (as described in Section 2.6.3.1) by the number of keys entered here.

If the size of the volume is specified in blocks by entering a negative number (e.g., -200), the number of keys available for the base directory is computed by dividing the specified key size into the number of blocks to be allocated.

2.6.4.2 Directory Extension Volume

BUILDPF requests the volume size for a directory extension volume

```
BASE VOLUME bb AND ITS CURRENT EXTENSIONS HAVE A TOTAL OF nnn BLOCKS
WHICH HOLD A MAXIMUM OF APPROXIMATELY kkkk KEYS.

NEW MAXIMUM NUMBER OF INDEXES (KEYS) [NEGATIVE FOR SIZE IN BLOCKS]:
```

Enter the number of keys to represent a new maximum number of keys for the base directory and its directory extensions combined (i.e., must have a value greater than kkkk). The new value is used to compute the total number of blocks needed to hold the keys. The difference between the new computed total blocks and kkkk is used to determine the size of the new volume. If the number of keys specified does not exceed the previous maximum (i.e., value given in kkkk), the prompt is redisplayed.

If the size is given in blocks, the size applies only to the volume being defined.

2.6.4.3 Data Volume

To determine the size of a data volume, BUILDPF requests the total number of records for that volume or the number of blocks required.

```
VOLUME SIZE IN RECORDS [NEGATIVE FOR SIZE IN BLOCKS]:
```

Enter a positive number to specify the number of records desired. BUILDPF then calculates the number of blocks required based on the number of records and record size.

Enter a negative value to specify the size of the volume in blocks. The number of blocks may be adjusted upward if the record size extends over block boundaries.

NOTE

Record size is the first entry when a polyfile is created.

2.6.5 BUILDING AND STRUCTURING VOLUMES

The building and structuring of polyfiles is the critical phase of BUILDPF. A malformed polyfile may result if an <ESC> or <CTRL-C> is pressed during this process. The beginning of the phase is indicated by the message

```
ALLOCATING VOLUME.  PLEASE WAIT.
```

When allocation is complete, BUILDPF displays

```
VOLUME xx ALLOCATION COMPLETE.
```

Structuring then takes place. This is indicated by one of the following messages:

```
STRUCTURING VOLUME xx AS BASE DIRECTORY VOLUME. PLEASE WAIT.

STRUCTURING VOLUME xx AS A DIRECTORY EXTENSION. PLEASE WAIT.

STRUCTURING VOLUME xx AS A DATA VOLUME. PLEASE WAIT.
```

where
xx - the assigned volume number

After a base directory volume or directory extension volume has been built and structured, BUILDPF prompts

```
EXTEND BASE VOLUME bb MORE ["Y"/"N"; <RETURN> = exit]:
```

If Y is entered, BUILDPF assumes volume type E (directory extension) for base volume bb (see Section 2.6.3 for volume type input). If N is entered, the user is given the option to build another type volume. To exit the program, press <RETURN>.

When structuring is complete, BUILDPF displays the messages

```
STRUCTURING COMPLETE.

LOGICAL UNIT (NONZERO) FOR VOLUME [<RETURN> = EXIT] :
```

At this point, <ESC> or <CTRL-C> may safely be used.

2.6.6 POLYFILE EXTENSION MODE

If the specified file has been previously created, BUILDPF enters extension mode. Volumes may then be added or an unstructured volume may be structured.

BUILDPF prompts

```
LOGICAL UNIT (NONZERO) FOR VOLUME  [<RETURN> = EXIT]:
```

The LU number entered must be in the range of 1 to 127.

For the volume number of the new volume or the volume number of an existing but unstructured volume, BUILDPF prompts

```
VOLUME NUMBER [0-63; <RETURN> = don't care]:
```

It is recommended that the default <RETURN> be used to allow BUILDPF to choose the lowest available volume number for the volume to be added or structured.

BUILDPF then requests volume type, as described in Section 2.6.3, and volume size, as described in Section 2.6.4, before building and structuring the volume.

Pressing <RETURN> exits the program and the system command prompt (#) is then displayed.

2.7 BUILDXF

BUILDXF is a BASIC program used to build indexed files. The program is entirely interactive. BUILDXF prompts for the parameters needed to build an indexed data or directory-only file as follows:

```
DESIRED FILENAME?
NUMBER OF DATA RECORDS?
DATA RECORD LENGTH (#WORDS)
NUMBER OF INDEXED RECORDS
NUMBER OF DIRECTORIES
ENTER KEY LENGTH (#WORDS) FOR EACH DIRECTORY:
#1?
```

- Number of data records - total number of actual data records required to be in the same file with the directories. The records are put on the free list. A zero may be entered if a directory-only file is to be built.

- Data record length (#words) - record length in number of words for the data portion of the file.

- Number of indexed records - maximum number of keys for the directory that is to contain the most keys. If the maximum number of keys to be contained in each of the different directories varies considerably, other indexed files may be built to contain the directories to avoid wasting disk space. This requires that one channel be used per indexed file.

- Number of directories - total number of directories required.

- Key length (#words) for each directory - key length in words for each directory. The length may differ for each directory.

After this information is entered, BUILDXF calculates space requirements for the directories, builds a contiguous file of the appropriate size, and then uses BASIC's SEARCH statements to set up all directories as specified.

The number of available data records is displayed. That number may be larger than the number specified due to internal file structure variations.

If there is insufficient contiguous space on the logical unit to build the file, the system displays an appropriate message. Contiguous space may be created by one of the following:

- Deleting a contiguous file of equal or greater size (deleting a text or formatted file may not create the required contiguous space). The space does not become available until the channel is closed by the last user accessing the file.

- Running CLEANUP as described in the IRIS R8 Operations Manual.

- INSTALLing AND CLEARing a new logical unit as described in the IRIS R8 Operations Manual. This deletes all files and accounting information on the LU.

2.7.1 SUMMARY OF INDEXED FILE FEATURES

An indexed file contains a number of blocks that serve as directories pointing to records contained in its data blocks. A maximum of 15 directories is allowed.

A directory-only file consists only of directories that may be used as pointers to records contained in other files. A directory-only file may also be used as a self-contained index. Thus, the number of directories that index the same data is almost limitless.

Each directory of an indexed file consists of three levels:

- Master
- Coarse
- Fine

Each level contains pointers to the next level and the fine level contains pointers to the associated record number. Thus, records are accessed by first searching the master level, then the coarse level, and finally the fine level.

The master level is always one disk block in length. The sizes of the coarse and fine level directories depend on the maximum number of data records in the file. Records located at the beginning of the file hold the directories and cannot be accessed directly via READ and WRITE statements. The first real data record and its record number depends on the number of records required for the directory.

The key length, which may be up to a maximum of 30 bytes, determines how many keys are contained in a block.

Number of keys per block may be calculated as

$$N=254/(K+1)$$

where
N - number of keys
K - key length

Since the master level is always one block, it may contain N keys. Each key points to a block on the coarse level. Theoretically, a directory could point to N^2 blocks in the fine level and to N^3-1 data records (one dummy key is inserted by the system at the end of each level).

The fine directory must contain 2R/(N+1) blocks, where R is the number of data records to be indexed. The coarse directory must contain F/(N-1) blocks, where F is the number of blocks required in the fine level.

The number of blocks must must be rounded up if a fraction results from the calculation. The number of blocks in the coarse level cannot exceed N since that is the maximum number of keys in

the master level. The minimum for the fine and coarse levels is two blocks each.

The number of blocks available on the logical unit and those used for the directories control the maximum number of data records in an indexed file.

A data-only indexed file cannot be built. To use the indexed file's free list capability, a minimum-sized directory consisting of five blocks must be built even if the directory is never used.

Refer to the IRIS Business BASIC Manual for more information on indexed files under IRIS.

Table 2-1 is a guide for calculating the maximum number of data records available in a given indexed file.

TABLE 2-1. MAXIMUM KEYS AND DATA RECORDS IN AN INDEXED FILE

Key Length (number of words) (K)	Keys Per Block (N)	Maximum Number of Keys in One Directory (R)
1	127	65534*
2	84	65534*
3	63	65534*
4	50	62475
5	42	37023
6	36	23310
7	31	14880
8	28	10962
9	25	7800
10	23	6072
11	21	4620
12	19	3420
13	18	2907
14	16	2040
15	14	1680

*This is the maximum number of records in any file except polyfiles.

2.7.2 USING BUILDXF

To use the BUILDXF command, at the system prompt (#), enter

```
BUILDXF
```

The following is an example of building an indexed file (user input is underlined):

```
PROGRAM TO CREATE AN INDEXED DATA FILE

DESIRED FILENAME? INDXTEST
NUMBER OF DATA RECORDS? 100
DATA RECORD LENGTH (#WORDS)? 8
NUMBER OF INDEXED RECORDS? 100
NUMBER OF DIRECTORIES? 3
ENTER KEY LENGTH (#WORDS) FOR EACH DIRECTORY:
#1 ? 3
#2 ? 4
#3 ? 5

PLEASE WAIT . . .

FILE HAS 100 DATA RECORDS

PLEASE WAIT . . .

FILE STRUCTURE COMPLETED

READY
```

When the system has finished structuring the file, the program chains to BASIC. Press <CTRL-C> to return to the system command prompt (#).

The QUERY command (see Section 2.22) may be used to look at the file. Information similar to the following is then displayed:

```
"INDXTEST" IS A CONTIGUOUS DATA FILE WITH 194 RECORDS OF 8 WORDS EACH

HBA: 0/4040,   TYPE:77032,  PRIV: 3,  CSIZE: 3130
PRIV: 2,  ACCOUNT GROUP: 1,   USER: 2, UNIT: 0
PROTECTION: 77,  SIZE: 47 BLOCKS,  AGE: 0 HOURS
LAST ACCESSED:  0 HOURS AGO,   COST:   $0.00,  TOTAL INCOME:  $0.00
```

2.7.3 USING BUILDXF FOR A DIRECTORY-ONLY FILE

The procedure is the same as shown in Section 2.7.2, except the response to the Number of Data Records? prompt should be zero.

2.8 BYE (LOG OFF)

BYE is the system command used to log off. It automatically updates the user account for the following:

- Accrued charges
- CPU and connect time
- Blocks in use
- Blocks available

At the system command prompt (#), enter

```
BYE
```

The terminal may display some or all of the following information:

```
#BYE   GROUP n USER nn  mon dd, yyyy  hh:mm:ss

NET ACCRUED CHARGES      $$$.cc

CPU TIME USED       n:nn:nn
CONNECT TIME USED   n:nn:nn

nnnnnn BLOCKS IN USE, nnnnnn AVAILABLE ON UNIT #n
```

2.9 CHANGE

CHANGE is a system command used to change certain file characteristics. Such changes may be made by the user who created a given file, users who have write-access to the file, or the system manager. The general user may change the following:

- Filename
- Cost
- Protection
- Locked (BASIC programs)

If the CHANGE processor is invoked from the manager account, the following may also be changed:

- R, L and I control bits
- Processor type
- File's starting address

2.9.1 CHANGING FILE CHARACTERISTICS

CHANGE prompts for each characteristic to be changed. If no change is to be made for a particular characteristic, press <RETURN>. The next prompt is then displayed.

CHANGE may be terminated after the desired changes have been entered by pressing <ESC>. Do not press <ESC> until the program has responded to the last <RETURN>. For example, if only the filename is to be changed, press <ESC> when the prompt, NEW COST?, is displayed.

2.9.1.1 Change Filename

To invoke CHANGE, at the system command prompt (#), enter

CHANGE filename

where
filename - name of the file to be changed

If the file is not on your assigned logical unit, the name must be entered in the form

lu/filename

where
lu - number of the logical unit on which the file resides

If the file cannot be changed because it is protected, an appropriate error message is printed; otherwise, the first prompt is

NEW NAME?

Type a new filename if desired. Do not include the logical unit number because the system defaults to the logical unit where the file was found. If the filename is to remain the same, press <RETURN>.

2.9.1.2 Change Cost

CHANGE displays the current cost (charge to others for access to the file) and prompts for a new cost

COST = $0.00 (or the rate which was set previously) NEW COST?

Enter the new cost or, if no change is desired, press <RETURN>. Note that cost is calculated in 10-cent increments.

2.9.1.3 Change Protection

CHANGE displays the current protection code and prompts for a new protection

```
PROTECTION = 77
NEW PROTECTION?
```

Enter a two-digit number or a zero to replace the current protection, or press <RETURN> to leave the protection unchanged. For information on the protection code, refer to Section 1.4.2.

If the file is a BASIC program, the program's locked status may be changed, as described in Section 2.9.1.4. If the user is on the manager's account, the manager options are displayed as described in Section 2.9.2. Otherwise, after the protection has been changed or bypassed, CHANGE exits and the system command prompt (#) is displayed.

2.9.1.4 Locking a BASIC Program

This option allows a BASIC program to be locked into a partition and prevents it from being swapped out until an input statement is encountered in the program code or the program has completed its task. The prompt appears only if the file is a saved BASIC program. For example, assume that seven user programs are enqueued for three partitions.

- Programs 1 and 2 each used their time slice and are swapped out.
- Program 3 is lockable, and once in the partition remains active until an input statement is detected or the program completes.
- Programs 4 through 7 and the requeued programs 1 and 2 now have only two partitions available to them and time-slice allotment slows down appreciably. Furthermore, the locked program may call another module, which then has to wait a long time for a time slice resulting in even greater delays.

POINT 4 recommends that this option be used only with the utmost discretion.

CHANGE displays the prompt

```
PROGRAM IS NOT LOCKABLE
LOCKABLE OR NOT (L OR N)?
```

Or the first line may say

```
PROGRAM IS LOCKABLE
```

In either case, if no change is desired, press <RETURN>. Otherwise, enter L (lockable) or N (not lockable).

2.9.2 MANAGER OPTIONS FOR CHANGE

Manager options for CHANGE allow a processor to be assembled and then changed, thus making it accessible for system use.

Depending on the type of file, a user signed on to the manager account has access to the following extended options:

- Change control bits
- Change processor type
- Change a file's starting address

When a filename is entered in response to the system command prompt (#), the file's header is read. The control bit in the header's TYPE word is then examined. Figure 2-1 shows control bits in the file header. If the R bit is set, the file becomes the port's selected processor.

The L and I bits of the file header are examined next. If the L bit is set, the system checks whether there is a qualifying filename associated with the processor's name. If such a filename is found, its header is read and the file is compared with the type of the processor. A match causes the program file to be loaded into the port's active file. The I bit is then checked (whether L was set or not) and input is enabled if I is set. When input is terminated, or if the I bit is not set, the selected processor is given control at its initial entry point.

If the R bit is not set, the system examines the file type. For a BASIC program, the proper processor (e.g., RUN filename is substituted for "filename") is selected.

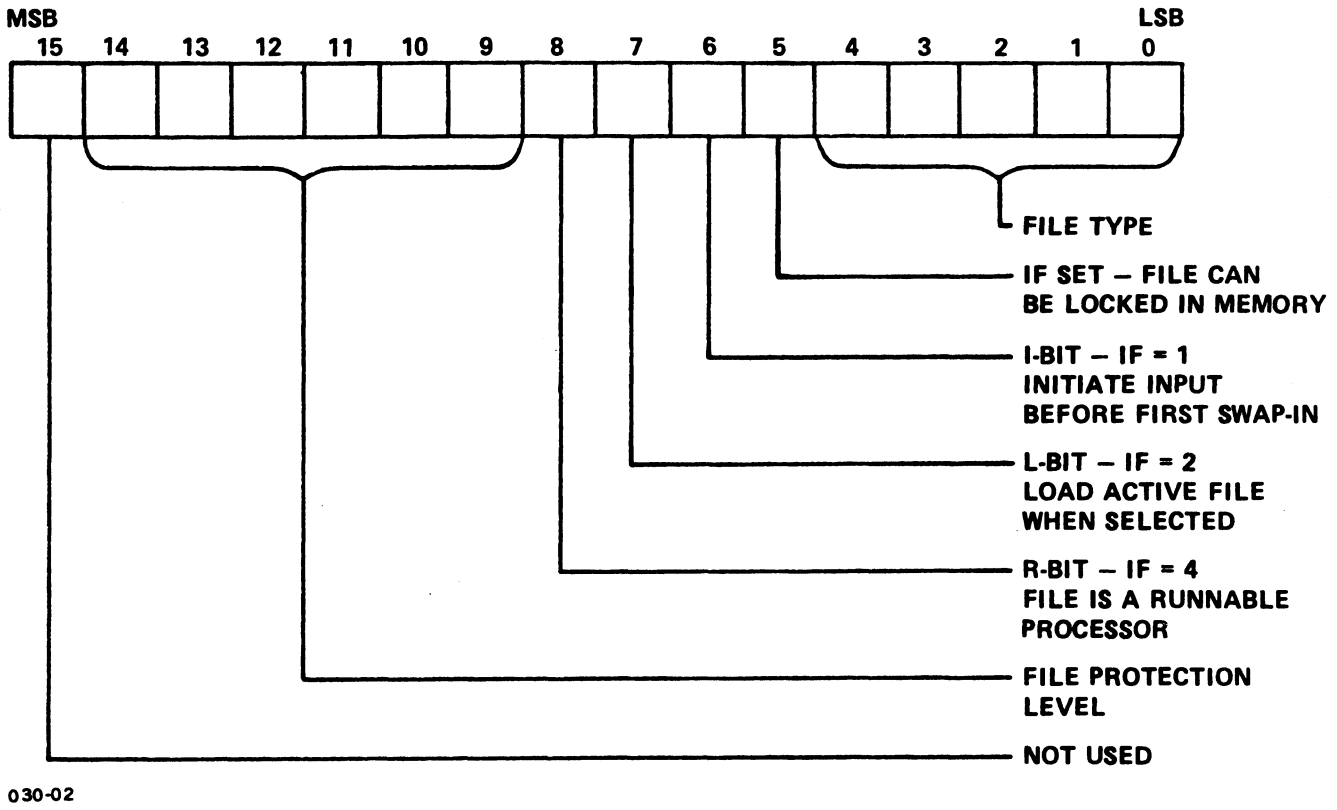

Figure 2-1. Control Bits in the File Header TYPE Word

2.9.2.1 Changing Control Bits

This option permits a change of control bits and CHANGE displays

```
R, L, AND I CONTROL = n
NEW CONTROL DIGIT?
```

where

n - control digit consisting of the third digit from the right of the file's TYPE word and is made up of three bits:

Value	Bit	Description
R=4	8	Runnable processor
L=2	7	Load active file
I=1	6	Initiate input

CHANGE allows the R bit to be set only if:

- First address in the processor is 200
- Processor will not overlay locations B0 through BPS-1
- File type is 1 or 3 (system or stand-alone file)

2.9.2.2 Changing Processor Type

If the R bit is set, CHANGE displays

```
PROCESSOR TYPE = x NEW TYPE?
```

This permits changing the file type (last two digits of the file header's TYPE word, see Figure 2-1) to match a processor with its program files. The least significant five bits of each file's TYPE word define the actual file type for classification purposes. Table 2-2 provides file type codes.

TABLE 2-2. FILE TYPE CODES

No.	Letter	File Type
0	P	Permanent system file
1	S	System processor or file
2	B	IRIS BASIC processor or program
3	A	Stand-alone processor or program
4-5		Reserved
6	M	SMBASIC processor or program
7-27		Reserved
30	T	Text file (ASCII)
31	F	Formatted data file
32	C	Contiguous data file
33		Reserved
34		Reserved
35		Reserved
36	$	Peripheral device driver
37		Reserved*

*A file type 37 may be specified by a processor to allow opening data file or data handler (types 30-36).

2.9.2.3 Changing a File's Starting Address

If the R bit is not set, CHANGE displays

```
STARTING ADDRESS = nnnnn
NEW STARTING ADDRESS?
```

where
nnnnn - current starting address in octal

If the file type is 3, a new starting address may be entered in octal. The starting address may range from 0 through 77777 (1nnnnn indicates no starting address).

Processors do not require a starting address as the system transfers control to a processor starting at BPS+4 (the fourth location after beginning of processor storage as defined in the IRIS Software Definitions).

2.9.3 CHANGE AND POLYFILES

CHANGE should not be used to change polyfile names. Use COPYPF.

2.10 COPY

COPY is a general purpose command for moving data of any type from a specified source, or several sources, to a specified destination. The logical unit specified for the destination file must have enough free blocks to contain the new file.

COPY is used for the following purposes:

- Copy one file
- Copy a file from one logical unit (LU) to another LU
- Merge several files into a single file
- Create a disk file from paper tape
- Punch a paper tape from a text file
- Compare and verify data files
- Page or depage a text file
- Copy text file(s) to a printer

NOTE

COPY should not be used on polyfiles. Refer to COPYPF for the correct procedure.

The COPY command format, including the most frequently used options, is

```
COPY {<pp> 1/}dest{!}={2/}source
```

where

```
    <pp> - new protection level
      1/ - destination logical unit
    dest - destination file
       ! - permits overwriting an existing file of the same type
      2/ - source logical unit
  source - file to be copied
```

If a protection level is not specified, the system defaults to protection level 77. For information on protection level codes, refer to Section 1.4.2.

2.10.1 COPYING ONE FILE ON A USER'S ASSIGNED LU

To copy one file, at the system command (#), enter

```
COPY {<pp>} dest = source
```

where
<pp> - new protection level
dest - name of the new file
source - name of the file to be copied

If the destination file is to replace an existing file, its name must be given in the form

```
dest!
```

After the file has been copied, the system displays the message

```
COPIED!
```

2.10.2 COPYING A FILE FROM ONE LU TO ANOTHER LU

If it is necessary to copy a file from one LU to another, at the system command (#), enter

```
source 4/dest11 = 0/source1
```

In this example, a file from LU 0 is copied to LU 4. If a file is to be copied from another LU to the user's assigned LU, the system default is used

```
COPY dest = lu/source
```

where
lu - logical unit number on which the source file is resident

2.10.3 COPYING CONTIGUOUS OR INDEXED DATA FILES

When copying a contiguous file or an indexed file, the destination file must have enough contiguous space to accommodate the source file. If the logical unit does not have enough contiguous space, an appropriate message is displayed. Contiguous space may be created by the following means:

- Delete an old contiguous file of equal or greater size. Deleting a text or formatted file may not create the required contiguous space.
- Run CLEANUP as described in the IRIS R8 Operations Manual.
- Create a new logical unit by running INSTALL AND CLEAR as described in the IRIS R8 Operations Manual.

At the system command prompt (#), enter

COPY {[r:w]} dest{!} = source

where
r - number of records
w - record length in words
! - overlays an existing contiguous file

If a new destination file is to be created, the number of records must be greater than or equal to that of the source file. The record size (number of words) must equal that of the source file. The number of records and record size ([r:w]) must be specified if the number of records and record size are to be increased. If they are to remain the same, the system defaults to the existing number and size, and these parameters need not be specified.

If the new file (i.e., dest) is built with more records than the source file, the additional records are not added to the free list. A BASIC program should be written to add those records to the free list.

2.10.4 CONCATENATING SEVERAL FILES

Multiple sources may be combined into a larger module in cases where it makes sense to concatenate the data. The following groupings are allowed:

1. ASCII sources (text files and byte-oriented peripherals, such as paper tape)

2. Binary sources (machine code files such as processors, binary format paper tape); these must be specified in ascending address sequence

Several text files may be combined into a single large file by entering

```
COPY filename = file1,file2,...
```

where
filename - new combined file
filel... - source files to be combined

Names of source files must be separated by commas.

The command may be rejected for one of the following reasons:

- Destination filename is already in use.
- Destination and one or more source files are not of the same type.
- Source file does not exist or is read-protected.

If an existing file is to be overlaid, enter the command in the format

```
COPY filename! = file1,file2,...
```

The source files (filel, file2, etc.) are not disturbed by use of the COPY command. However, since all of the text has been reproduced in the combined file, the original source files may be deleted if desired.

2.10.5 CREATING A FILE FROM PAPER TAPE

A text file may be created by loading ASCII-coded paper tapes into a high-speed paper tape reader or the terminal's paper tape reader. Other types of paper tapes may also be read into a file via the high-speed paper tape reader. In either case, the file type must be supplied if the first source is a peripheral device. See Table 2-2 for file type codes.

If the destination is a text file, COPY checks:

- All characters read from tape for even parity; all null codes (blank frame), rubout codes, and linefeed codes are ignored
- Every 50 characters for lost characters; if the leader is followed by a null, each 50th character thereafter must also be null; an error is indicated if the nulls are not found; this provides for the easy detection of characters that may have been lost in transmission

To read a paper tape into a text file, at the system command prompt (#), enter

```
COPY filename*T = $PTR
```

If more than one tape is to be read into the file enter the command in the form

```
COPY filename*T = $PTR <UP-ARROW> $PTR <UP-ARROW> $PTR
```

This permits three tapes to be read into the destination file. The up arrow before the source name will cause COPY to wait until the tape is loaded into the reader. COPY signals when it is ready for another tape by displaying the prompt

```
LOAD DEVICE THEN PRESS RETURN
```

After the new tape is loaded into the reader, pressing <RETURN> allows COPY to continue.

2.10.6 CREATING A FILE ON PAPER TAPE

To punch a tape on a high-speed paper tape punch, at the system command prompt (#), enter

```
COPY $PTP = filename
```

where
filename - file to be punched

The entire file is punched with even parity. Leader and trailer are also punched, and a null (blank frame) is punched after the first byte and every 50th byte thereafter.

COPY uses these nulls to check for characters being picked up or dropped by the paper tape equipment.

If the peripheral device cannot accept the data from the tape, its driver generates an error indicated by

```
ITEM TYPES DON'T MATCH
```

2.10.7 COPYING ANY TYPE OF DATA FORMAT TO PAPER TAPE

Some data types not known to the system may be transferred from one byte-oriented peripheral device to another by using the image mode COPY command given in the form

```
COPY dest _ source
```

where

_ (underscore)	- signifies that the destination is to be a direct copy of the source without regard to the data type

2.10.8 COMPARE AND VERIFY DATA

The COPY command can be used to compare or verify data from one file or device against the data from one or more other sources by typing a question mark ahead of the destination filename.

Enter the command in the format

```
COPY ? dest = source1,source2,...
```

If a discrepancy is detected, an appropriate error message is displayed.

NOTE

Compare and verify cannot be used on indexed files.

2.10.9 PAGE OR DEPAGE A TEXT FILE

The COPY command can be used to page or depage a text file.

At the system command prompt (#), enter

```
COPY dest#n = source1,source2,...
```

where
n - desired number of lines per page; if n is omitted, COPY defaults to 50 lines per page

If the source is a peripheral device such as $PTR, any existing formfeed codes in the source text are deleted and replaced by a formfeed code each nth line by using the command

```
COPY dest*T#n = source1,source2,...
```

where
n - number of lines (maximum is 65535)

To depage a text file (remove all formfeed codes), a similar command may be issued with a value for n that is larger than the number of lines of text. Since the maximum value is 65535, this is a safe value to use for depaging.

If BASIC is used to load a file, it is not necessary to depage it because BASIC ignores form codes in the source text.

2.11 COPYPF

COPYPF is a BASIC program that provides an efficient method for copying polyfile volumes from one logical unit to another. Polyfile integrity is maintained throughout the process. COPYPF may be used for the following purposes:

- Copy one polyfile
- Copy many polyfiles in one jobstream
- Copy selected volumes of a polyfile
- Change the filename
- Recover a polyfile from a backup disk

COPYPF selects files based on parameters entered by the user. It may be used to copy a master volume and/or selected volumes of a particular polyfile. If the master volume (volume 0) is copied, all volumes in that polyfile must be copied.

The initial selection module requires only the name of the polyfile to be copied and the location (i.e., logical unit number) of volume 0. Destination logical units for individual volumes are specified in a subsequent module.

The selected polyfile names are displayed in groups of 36. Each group constitutes a page. Only those files that have been reviewed by the user after initialization are copied. Files listed on pages that the user does not review are not copied. A copied file exists both on its original and its new logical unit.

NOTE

It is recommended that QUERYPF be run to ascertain the current polyfile status (i.e., whether it can be opened and that the source logical unit is installed) before invoking the COPYPF program.

2.11.1 COPYPF WORK FILES

COPYPF builds three work files (EDITSV0ppp, EDITSWKAppp, and UTYPRCppp) to accumulate lists of specified filenames for copying. Files listed in the EDITSV0ppp file are displayed on the terminal screen for the user to review and modify. The selected filename(s) are then written to EDITSWKAppp. Information about the selected file(s) is written to UTYPRCppp. Each user is assured access to a unique set of files because the program incorporates the the user's port number (represented by ppp) into the work filenames.

2.11.2 COPYPF HELP MODULES

Two help modules are available. The first guides the user in the selection of file parameters. The second help module contains detailed descriptions of COPYPF modification commands.

Press <RETURN> to exit the help module and return to the parameter prompts.

2.11.3 COPYPF PROCEDURE

The following is a step-by-step description of the COPYPF procedure.

1. If COPYPF is resident on either LU 0 or the user's assigned LU, at the system command prompt, (#) enter

   COPYPF

   If the program resides on another LU, enter the command in the form

   lu/COPYPF

   where
   lu - number of the logical unit on which the program resides

   While the program builds its work files, it displays

   FILE INITIALIZATION IN PROGRESS, DO NOT DISTURB!

   When the work files have been built, the following selection menu is displayed:

```
FILE SELECTION     POLYFILE COPY FACILITY     COPYPF 1.0  mm/dd/yy


                    FILE(S) PREFIX: _____________

                  SOURCE LOGICAL UNIT: __

                       REPORT OUTPUT: _

                  OPTIONAL PARAMETERS: _____________


COMMENT: ENTER A QUESTION MARK (?) AT ANY TIME FOR HELP
COMMAND:
MESSAGE:
```

   The top line of the screen shows the user's port number, program name, the program's version, and release date. File selection prompts are in the center of the display.

   The user may enter the first help module by entering ? in the first keyable position of any field. Explanations are given for the various options permissible for the parameter entries.

   Pressing <ESC> at an entry field aborts the current entry and returns to the previous field. <ESC> at the first field aborts the COPYPF program.

The three lines at the bottom are for system/user interaction. COMMENT provides system prompts for the user, COMMAND is for user input, and MESSAGE displays messages from the system.

File selection prompts require the following parameter entries:

- FILE(S) PREFIX - allows the user to specify the beginning characters of the filenames. This is particularly helpful when naming conventions have been used to group files into categories. Only polyfiles are selected by COPYPF regardless of the specified prefix.

  If only one polyfile is to be processed, enter its filename. If a group of polyfiles are to be processed, use the common prefix characters of the filenames. A <RETURN> defaults to all polyfiles on the specified or default source logical unit.

- SOURCE LOGICAL UNIT - specify the LU where volume 0 (the master volume) of the polyfile resides. A <RETURN> defaults to the user's assigned LU.

- REPORT OUTPUT - specify the system printer or a file that may be used to display the information at a terminal or print the report at a later time. To print the report on a system printer, enter a number (0-9 for $LPT-$LPT9). If this option is selected, a report is printed upon completion of the COPYPF processing.

  To create a file, enter **F**; the program prompts

  REPORT OUTPUT FILENAME: _____

  Enter a unique filename of up to 14 alphanumeric characters. The filename must follow the standard IRIS naming convention.

  The report gives the following information:

  - Polyfiles that were copied
  - Their source and destination LUs
  - New filename(s)
  - COPYPF status information

  COPYPF uses IRIS Business BASIC CALL 91 to perform the required functions. If processing was unsuccessful, the report explains why. A CALL 91 status list is provided in Section 2.23.4. If the specified printer is not available, the report may be written to a file created by the user. This file may be displayed at the terminal or printed at a later time. Sample reports are shown in Section 2.11.5.

If no report is required, enter N or press <RETURN> (N is the default).

- OPTIONAL PARAMETERS - enter any valid LIBR command except the LIBR * command.

2. After the parameters have been selected with valid or default entries, the program displays the message

```
IS ALL THE ABOVE CORRECT ?
```

If N is entered, the cursor is returned to the optional parameter prompt.

To begin the file selection process, enter Y. The program chains to the LIBR processor and displays

```
LIBR * C ^ _ [<00>EDITSVnn!]

BUILDING WORK FILE, DO NOT DISTURB!
FILE INITIALIZATION IN PROGRESS, DO NOT DISTURB!
FORMATTING WORK FILE, DO NOT DISTURB!
```

3. When the work file is built and formatted, the program displays the following review menu, listing the specified files in groups of 36.

```
FILE REVIEW    POLYFILE COPY FACILITY        COPYPF1  mm/dd/yy

LOGICAL UNIT# nnn

NO.  *  FILE NAME      NO.  *  FILE NAME      NO.  *  FILE NAME

==   =  =========      ==   =  =========      ==   =  =========
01.  C  FS10FD@         13.  C  ROUTINPUT@
02.  C  FS10FH@         14.  C  ROUTKILL@
03.  C  FS10FHD@        15.  C  ROUTNUMCHK@
04.  C  FS10FHE@        16.  C  ROUTOPEN@
05.  C  FS10FHF@        17.  C  ROUTPERCENT@
06.  C  FS10FHM@        18.  C  ROUTTRUNK@
07.  C  FS10FHP@        19.  C  ROUTWORD@
08.  C  FS10FHR@        20.  C  ROUTZFIL@
09.  C  FS10FHS@        21.  C  SCAN@
10.  C  FS10FHX@
11.  C  POLYDATA@
12.  C  POLYERROR@


COMMENT: A=ALL,E=EXECUTE,P=PAGE,R=RESTART,?=HELP,nn=FILE NUMBER
COMMAND:
MESSAGE:
```

If <ESC> is pressed any time after the first file list is displayed, the program aborts and control is returned to SCOPE.

When the first group of 36 files is displayed, the user may access the second help module. This module assists in the use of the review commands. For a summary of the review commands, see Appendix D.

The user should examine all pages of the selected files before entering the E (Execute) command. Files listed on pages that are not reviewed are not processed.

The W (Wrap) command allows the user to check files selected for processing. The W command renumbers those files that were not deleted during the previous selection process and displays them from the beginning. After the command is entered, every page must be reviewed again.

The polyfile selection is not complete until the review is finished and the E command is entered.

4. Following the E command, the program displays the volume/LU selection screen for further specifications as follows:

```
PORT NUMBER nn   POLYFILE COPY FACILITY   COPYPF2 1.0  mm/dd/yy

                NAME OF POLYFILE: (polyfilename selected)

VOL# LU#  VOL# LU#  VOL# LU#  VOL# LU#  VOL# LU#  VOL# LU#  VOL# LU#
==== ===  ==== ===  ==== ===  ==== ===  ==== ===  ==== ===  ==== ===

COMMENT:
COMMAND:
MESSAGE:
```

The name of the first polyfile selected for copying is displayed at the top of the menu. After the destination LU and/or the destination filename is entered, only those volumes associated with that polyfile are shown on the Volume/LU Menu.

- Destination LU # - enter the new logical unit number for the polyfiles master volume (volume 0) if it is to be copied. If volume 0 is copied, all associated nonzero volumes must also be copied to update pointers to the master volume. The program aborts if the nonzero volumes are not copied.

  If the master volume is to remain on the same LU, but some volumes are to be copied to different LUs, press <RETURN>. The program displays the following message at the destination LU prompt

  NO COPY

  The cursor skips the destination file prompt and positions at the first nonzero volume associated with the polyfile. Selected nonzero volumes may then be copied to other LUs.

  NOTE

  If volume 0 is not copied to another LU but some or all of the associated nonzero volumes are copied, volume 0 is updated to point to the copied nonzero volumes. The original volumes residing on the source LUs have lost their master volume and are no longer accessible. They should be deleted as individual files using the KILL command.

  If the polyfile displayed is not to be copied, press <ESC> to proceed to the next selected polyfile. If <ESC> is pressed when the displayed file is the last of the selected files, the program assumes all specifications are complete (see step 6).

- Destination filename - enables the user to change the name of the polyfile during the copy process. Enter the new name. If the name entered does not end with the at sign (@), it is automatically appended. The new name is then assigned to each volume copied. If the name is not to be changed, press <RETURN>.

5. Following a valid entry for destination file (or if the prompt was skipped) the volumes for the displayed polyfile are listed in the "VOL# LU#" columns.

   Enter the desired logical unit number (i.e., any LU number from 1 to 127 that is installed) for the volume to be copied. The program checks whether the specified LU has enough contiguous space to accommodate the volume. If the name is changed, it checks whether the new name is in use on the destination LU.

   If volume 0 was copied, all associated volumes must also be copied. If volume 0 was not copied, the user may select which nonzero volumes are to be copied. A <RETURN> at any volume/LU entry indicates that the volume is not to be copied.

   The program then asks

   IS ALL THE ABOVE CORRECT?

   Enter Y and the shells for the volumes are built while the program displays:

   DO NOT DISTURB, ALLOCATING VOLUME "nn" OF "filename@"

   No data is entered into any volume until the file transfer is initiated (see step 6); however, the master volume is updated to contain the LU number and disk address for each volume.

6. After the specifications are complete for all volumes of the displayed polyfile, the process is repeated until the last of the selected files is displayed and the specifications are completed. The program then asks

   DO YOU WISH TO PROCEED WITH THE FILE TRANSFERS?

   A Y response initiates the copying of all the specified polyfiles and their volumes while the program displays the message

   00000 DO NOT DISTURB,COPYING VOLUME -> "nn" OF "file@"

   The message repeats for each volume processed. The numbers to the left of the message constitute a counter that increments every 10 blocks.

   The program writes a report on the transfers into a special file and the program displays

   PRINTING REPORT PAGE NUMBER n

   The page number is incremented as required.

If N is entered, a warning message is displayed

```
WARNING:  YOU MUST ENTER Y IF (NO COPY) WAS SPECIFIED ON VOL 0
```

The transfer prompt is then redisplayed.

If N is reentered, the files created on the destination logical unit(s) are maintained but no data is written into them. The files should be deleted using KILLPF (see Section 2.16) unless the user wishes to copy the polyfile structure instead of using BUILDPF.

NOTE

If only selected volumes are to be copied (i.e., volume 0 was not copied), do not enter N. If N is entered, the master volume will point to the copied volumes, the original volumes are inaccessible, and the file must be recovered from a backup disk.

7. If a report was requested, it is displayed or printed upon completion of the COPYPF process. The report shows whether the volumes of a particular polyfile were successfully copied. Sample reports are shown in Section 2.11.5.

8. When processing of all the selected polyfiles and their volumes is complete, control is returned to system command mode.

```
NORMAL EXIT FROM COPYPF #
```

2.11.4 NOTES ON USING COPYPF

If COPYPF cannot find a polyfile that is known to exist, ask the computer operator whether any logical unit numbers were changed when the system was brought up (i.e., when INSTALL or INSTALL AND CLEAR was run). If the logical unit assignments were changed, ASSIGNPF must be run. ASSIGNPF is described in the IRIS R8 Operations Manual. This program resets the pointers from the master volume (volume 0) to the associated nonzero volumes. After ASSIGNPF is run, the COPYPF utility may be used.

An alternative method for finding a missing polyfile is to run QUERYPF (see Section 2.23).

2.11.5 SAMPLE COPYPF REPORTS

COPYPF reports show the status of each polyfile and its volumes. The report may show the successful conclusion of the copy process or it will indicate why and at what point it was aborted with one or more of the following messages:

COPY COMPLETED

POLYFILE OPEN FAILURE

USER REQUESTED SKIP COPY

SOURCE FILE MISSING

DESTINATION FILE MISSING

DISK SPACE UNAVAILABLE

RESERVED

INVALID FILE STRUCTURE

POLYFILE ERROR STATUS

Examples of some COPYPF reports are shown in Figure 2-2.

OCT 1, 1984 3:24:16 PAGE NUMBER - 001

POLYFILE COPY REPORT

POLYFILE TESTPF@ COPIED TO TESTPF@

VOLUME	SOURCE LU #	DESTIN LU #	COPY STATUS INFORMATION
00	006	007	COPY COMPLETED
01	006	007	COPY COMPLETED
02	006	007	COPY COMPLETED
03	006	007	COPY COMPLETED
04	006	007	COPY COMPLETED
05	006	007	COPY COMPLETED
06	006	007	COPY COMPLETED
07	006	007	COPY COMPLETED
08	006	007	COPY COMPLETED
09	006	007	COPY COMPLETED
10	006	007	COPY COMPLETED
11	006	007	COPY COMPLETED
12	006	007	COPY COMPLETED
13	006	007	COPY COMPLETED
14	006	007	COPY COMPLETED
15	006	007	COPY COMPLETED
16	006	007	COPY COMPLETED
17	006	007	COPY COMPLETED
18	006	007	COPY COMPLETED
19	006	007	COPY COMPLETED
20	006	007	COPY COMPLETED
21	006	007	COPY COMPLETED
22	006	007	COPY COMPLETED
23	006	007	COPY COMPLETED
24	006	007	COPY COMPLETED
25	006	007	COPY COMPLETED
26	006	007	COPY COMPLETED
27	006	007	COPY COMPLETED
28	006	007	COPY COMPLETED
29	006	007	COPY COMPLETED
30	006	007	COPY COMPLETED
31	006	007	COPY COMPLETED
32	006	007	COPY COMPLETED
33	006	007	COPY COMPLETED
34	006	007	COPY COMPLETED
35	006	007	COPY COMPLETED
36	006	007	COPY COMPLETED
37	006	007	COPY COMPLETED
38	006	007	COPY COMPLETED
39	006	007	COPY COMPLETED
40	006	007	COPY COMPLETED
41	006	007	COPY COMPLETED
42	006	007	COPY COMPLETED
43	006	007	COPY COMPLETED
44	006	007	COPY COMPLETED
45	006	007	COPY COMPLETED
46	006	007	COPY COMPLETED
47	006	007	COPY COMPLETED
48	006	007	COPY COMPLETED
49	006	007	COPY COMPLETED
50	006	007	COPY COMPLETED

OCT 1, 1984 3:24:32 PAGE NUMBER - 002

POLYFILE COPY REPORT

POLYFILE TESTPF@ COPIED TO TESTPF@

VOLUME	SOURCE LU #	DESTIN LU #	COPY STATUS INFORMATION
51	006	007	COPY COMPLETED
52	006	007	COPY COMPLETED
53	006	007	COPY COMPLETED
54	006	007	COPY COMPLETED
55	006	007	COPY COMPLETED
56	006	007	COPY COMPLETED
57	006	007	COPY COMPLETED
58	006	007	COPY COMPLETED
59	006	007	COPY COMPLETED
60	006	007	COPY COMPLETED
61	006	007	COPY COMPLETED
62	006	007	COPY COMPLETED
63	006	007	COPY COMPLETED

COPY COMPLETED AS SPECIFIED

Figure 2-2. Sample COPYPF Reports

2.12 DISPLAY

DISPLAY is a BASIC program that allows the user to display a text file. The text file may not exceed 2000 lines. The program does not permit editing.

Twenty-three lines of text are displayed at a time. The various viewing commands allow the user to view the file forward or backward. The menu of viewing commands is displayed at the bottom of the screen.

To use DISPLAY, at the system command prompt (#), enter

DISPLAY {lu/}filename

where

lu - number of the logical unit on which the file resides; the logical unit number need be entered only if the file does not reside on the user's assigned logical unit

filename - name of the text file to be displayed

If the specified file is a text file, the first 23 lines of text are displayed. The menu of viewing commands (displayed at the bottom of the screen) is

```
E=End, U=Up, B=Beginning, (space)=Down, Z=EOF :      LINE  23
```

where

(space) - space bar

The cursor is positioned at the colon; the command code is entered without a <RETURN>. The following are viewing commands and their functions.

Command	Function
E	Exits the program and returns to the system command prompt with the message USER REQUESTED EXIT #
U	Displays 23 lines up (or back to the beginning of the file).
C	Displays 23 lines using the current first line as the center.
B	Displays the first 23 lines of the file.
(space)	Displays the next 23 lines.

Z Scrolls to the end of the file. Line numbers are incremented in multiples of 20 until the end of the file is reached. The number of the last line is then displayed as follows:

```
### EOF ###                    LINE nnn
```

where
nnn - number of the last line in the file

If the file is not a text file, cannot be found, or is too large, an error number is displayed. The program then chains to the BASIC help module and an appropriate message is displayed followed by the system command prompt.

For example, if the specified file is not a text file, the error number and message are

```
ERROR 44 AT LINE 142

HELP: NOT A DATA FILE (CAN'T OPEN OR REPLACE)
```

2.13 EXTRAPORT (PHANTOM PORT)

EXTRAPORT is a BASIC program that may be used to run selected programs on a phantom port. It can be used only from the manager account.

A phantom port differs from an interactive port only in that it has no terminal. A phantom port may be used to run a job in background while the terminal is used for other purposes (e.g., outputting a LIBR listing to a text file or a printer).

Under IRIS, two methods are available for using a phantom port: CALL 98 and EXTRAPORT. A CALL 98 may be inserted into a BASIC program. Please refer to the IRIS Business BASIC Manual for information on this procedure.

EXTRAPORT uses CALL 98 to initiate a job.

If the job runs successfully, the phantom port remains on the system. The program exits and the system command prompt (#) is displayed.

To run on a phantom port, at the system command prompt (#), enter

```
EXTRAPORT
```

If a phantom port is available, the program displays

```
!
```

Enter the program name or the system command of the desired processor; for example

```
LIBR @^[$LPT]
```

The program sends the command string to the selected phantom port, displays an appropriate message, and returns the user's port to system command mode; for example

```
LIBR IS RUNNING ON PORT 1
#
```

If all ports are busy, the message is

```
ALL PHANTOM PORTS ARE BUSY ! ! !
#
```

Wait a few minutes and then try again.

EXTRAPORT displays the following message if the command failed:

```
LIBR FAILED ! ! !
#
```

Failure may be due to an error in the command string. Repeat the procedure. If it fails again, report the problem to the system manager.

2.14 FINDFILE

FINDFILE is a BASIC program that is used to locate a specified file. The program searches each installed logical unit on the system and displays the number of the logical unit where the file resides.

To use FINDFILE, at the system command prompt (#), enter

```
FINDFILE
```

The following message is then displayed:

```
THIS PROGRAM WILL DISPLAY
LOGICAL UNIT NUMBER(S) WHERE
THE FILE IS LOCATED
```

At the prompt, FILE, enter the name of the file to be located. In the following example, the TESTX file is specified:

```
FILE: TESTX

NOW LOOKING

TESTX IS ON LU1  STILL LOOKING
TESTX IS ON LU/2  STILL LOOKING
TESTX IS ON LU/5  STILL LOOKING

ALL DONE!!!

    THIS PROGRAM WILL DISPLAY
    LOGICAL UNIT NUMBER(S) WHERE
    THE FILE IS LOCATED
FILE:
```

The user may enter another filename or press <ESC> to exit the program.

If the specified file is not found, the message displayed is

```
name NOT FOUND!!!
******************
```

2.15 FORMAT

The FORMAT command is used to create formatted or contiguous files. It may also be used to replace an existing formatted or contiguous file. The syntax of the command string may include the optional protection code and cost.

<ESC> may be used to abort the formatting procedure. The new file will automatically be deleted. If an existing file is being replaced, it is restored to its previous status.

2.15.1 FORMATTED DATA FILES

A formatted file consists of a file header and up to 32768 data blocks. Blocks are allocated to the file automatically as they are required. Each block (256) words can contain one or more records. Each record can contain up to 64 items (fields). Each item can be defined as an ASCII string, a decimal number, or a binary word field. The specified length and type of each item is recorded in a map contained in the file header.

2.15.1.1 Creating a Formatted File

To create a new formatted file, at the system command prompt (#), enter

```
FORMAT filename
```

where
filename - any legal filename

This command will be rejected if the filename is already in use on the specified logical unit (LU). An existing formatted data file on the user's account may be replaced by entering the command in the form

```
FORMAT filename!
```

If the file protection and cost is other than the default (pp = 77 and cost = $0.00) and if the file is to be built on another logical unit, enter the command in the form

```
FORMAT {<pp> $ddd.cc lu/}filename
```

where
<pp> - desired protection code
$ddd.cc - amount to be charged for access to the file by other users
lu - number of the logical unit where the file is to be built

For information on protection levels and cost, refer to Section 1.4.2.

FORMAT then prompts for information on the items to be contained in each record starting with the prompt

```
ITEM #0:
```

Codes are used to specify each type of item in a record.

Code	Description
S	String of ASCII characters. The letter S must be followed by a number specifying the dimension (maximum number of characters) of the string. If an odd number is given, the dimension will be the next higher even number.
F	Floating point binary number. This number form is not used by Business BASIC.
D	Decimal number. This is a binary-coded decimal number of the form used by Business BASIC. • D or D1 specifies a one-word decimal integer in the range ±7999). • D2, D3, and D4 specify two-, three-, and four-word floating-point decimal numbers, respectively.
B	Binary data. The letter B must be followed by a number specifying the dimension (number of words) of the item.

Item identifiers must be separated by commas. Several items of the same type and dimension may be specified by entering the number of identical items ahead of the letter. For example, to specify that the next three items are to be strings of 16 characters each, enter

```
3S16
```

To terminate input of items, press <RETURN> at the next prompt.

After all the desired items have been specified, FORMAT calculates the required record length based on the item information entered and displays the result

```
RECORD LENGTH = nn WORDS
```

2.15.1.2 Formatting Example

In this example, a formatted file named PAYROLL is created. It is to be write-protected against lower privilege-level users. Each record is 41 words long (i.e., 6 records per data block with 10 words unused in each block). Assume the following requirements:

Item	Description
#0	A string of 20 bytes
#1 - #4	Four 2-word decimal floating-point numbers
#5 and #6	Two strings of 14 bytes each
#7 and #8	Two 3-word decimal floating-point numbers
#9	A string of 3 bytes
#10	A 1-word decimal integer

The dialog with FORMAT (user input is underlined) is as follows:

```
#FORMAT <20> PAYROLL

ITEM #0: S20,4D2

ITEM #5: 2S14,2D3,S3,D1

ITEM #11: <RETURN>

RECORD LENGTH = 41 WORDS
```

When FORMAT requests input for item #11, <RETURN> is pressed to indicate that no further items are to be specified.

Note that item #9 is calculated by the system as a string of up to 4 bytes, although the user specified 3 bytes; this is because an integral number of words must be allotted for each item.

A formatted data file may also be created and formatted by use of a BUILD statement in a BASIC program. Refer to the IRIS Business BASIC Manual for information on the procedure.

2.15.2 CONTIGUOUS DATA FILES

A contiguous file consists of a header and a number of data blocks. It accommodates mixed record formats.

A contiguous file must have contiguous blocks allotted to it. These blocks must be allocated when the file is first built, whether all blocks will be used immediately or not.

A contiguous file can be converted to a polyfile volume provided there is no data in the file. The procedure for converting a contiguous file into a polyfile is described in the IRIS Business BASIC Manual.

2.15.2.1 Creating a Contiguous File

To build a contiguous file, at the system command prompt (#), enter

```
FORMAT {$ddd.cc <pp>} [r:w] {lu/}filename{!}
```

where

$ddd.cc <pp> - optional cost and protection parameters; for information on protection code parameters, see Section 1.4.2; the default for cost is $0.00 and for protection is 77

r - number of records the file is to contain; any positive integer up to 65534 is allowed, provided 1+(r*w/256) sequential disk blocks are available on the specified LU

w - number of 16-bit words per record; any positive decimal integer up to 32768 is allowed

lu/ - file is to be built on specified logical unit

! - permits replacement of an existing contiguous file (of equal or greater size)

If the logical unit has enough contiguous space available, the system displays

```
FILE SIZE = nnnn BLOCKS
#
```

If the logical unit does not have enough contiguous blocks, the system displays

```
LOGICAL UNIT DOES NOT HAVE ENOUGH FREE BLOCKS
#
```

Contiguous space may be created by one of the following means:

- Deleting an old contiguous file. Deleting a formatted or text file may not create sufficient contiguous space.
- Running CLEANUP, as described in the IRIS R8 Operations Manual.
- Creating a new logical unit by running INSTALL AND CLEAR as described in the IRIS R8 Operations Manual.

2.15.2.2 Example of Building a Contiguous File

In this example, a contiguous file named INDATA is to be created with protection 33 on logical unit 2. It is to contain 3000 records of 128 words each. There will be no charge to other users for access to the file.

```
FORMAT <33> [3000:128] 2/INDATA
```

2.16 KILL

One or more saved BASIC programs or data files may be deleted from a disk by using the system command KILL. The freed disk blocks are returned to the general pool and the owner's account is updated. Only the user's own file, or a file belonging to another user that is not write-protected, may be deleted.

U.KILL may be used if a large number of files are to be deleted.

2.16.1 USING KILL

To delete one file or program, at the system command (#), enter

```
KILL filename
```

If the file resides on a logical unit (LU) other than the user's assigned LU, enter

```
KILL lu/filename
```

where
lu - number of the logical unit

If the deletion process is successful, the message displayed is

```
DELETED
```

If the deletion process is unsuccessful, an appropriate message is displayed (see Section 2.16.2).

More than one file (which may reside on different LUs) may be deleted at the same time. Filenames must be separated by a comma or a space. The command format is

```
KILL filename1 filename2, lu/filename3
```

where
lu - number of the logical unit on which the file resides; it is not necessary to specify an LU when deleting files on the user's assigned LU

If the procedure is successful, the program displays

```
ALL DELETED
```

2.16.2 EXAMPLE OF USING KILL

In the following example, seven files are to be deleted; however, the first and fifth filenames (Testfile and JJ) do not exist on the user's LU. The second file (98#DO) is an illegal filename. File number 7 (2/Pay) is on an LU that is not installed. File 4/MAX is on another user's account and is write-protected. File COPY is a system file that cannot be deleted without first changing it to another file type.

```
KILL TESTFILE,98#DO,4/MAX,BJ.22,1/JJ,COPY,2/Pay
```

The messages displayed by KILL are as follows:

```
? LOGICAL UNIT NOT ACTIVE:  7
? FILE NOT FOUND:  1  5
? ILLEGAL FILENAME:  2
? FILE IS WRITE PROTECTED:  3
? FILE IS A PROCESSOR OR DRIVER:  6
```

The numbers refer to the sequence in which the filenames were entered into the command string.

In this example, only the fourth file (BJ.22) was deleted.

2.16.3 DELETING POLYFILES

Polyfiles need special handling and it is recommended that KILLPF (see Section 2.17) be used. However, if KILL is used, volumes must be deleted one at a time. Volume 0, the master volume, must be deleted last.

Run QUERYPF (see Section 2.23) to get the latest volume and LU information. With the listing at hand, at the system command (#), enter

```
KILL pfname@xx
```

where
xx - two-digit volume number

When a volume resides on an LU other than the user's own LU, enter the command in the form

```
KILL lu/pfname@xx
```

where
lu - number of the logical unit where the polyfile volume resides

Repeat this command until all volumes have been deleted ending with volume 0.

2.17 KILLPF

KILLPF is a BASIC program specially designed for the deletion of polyfiles. It deletes each volume starting with the highest numbered volume and deletes volume 0 (the master volume) last while reporting on each volume as it is deleted. The procedure is as follows:

At the system command prompt (#), enter

```
KILLPF
```

The program responds by acknowledging the command and asking for the polyfile name

```
KILLPF - Kill Polyfile Utility
Polyfile name [should have "LU/"]:
```

Enter the polyfile name; the dialog (user input is underlined) will be similar to the following example:

```
#KILLPF
 KILLPF - Kill Polyfile Utility
 Polyfile name [should have "LU/"]: 1/PFname@  Validating polyfile structure.
 Please wait...
 March 16, 1982  21:55:10
 Polyfile:  PFname@
 Volume 0 LU: 1  Total data records allocation: 0
 Record size is 256 words Total volumes: 3 Last accessed: 0.46 hours ago
 Vol LU/ NBLK Type | Vol LU/ NBLK Type | Vol LU/ NBLK Type | Vol LU/ NBLK Type
  0   1/  25   B 1 |  1   1/  24   E 0 |  2   1/  42   D M |  3

 Type "YES" to confirm deletion: YES
```

After YES has been entered, each volume is listed as it is deleted:

```
Volume 2  deleted
Volume 1  deleted
Volume 0  deleted
Polyfile deletion sequence complete
#
```

At the end of the sequence or if the response is NO, control is returned to system command mode.

NOTE

If a volume is marked by an asterisk (i.e., the LU is not installed), the prompt to confirm deletion is repeated. Before the delete process can continue, the LU must be installed.

2.18 LIBR

LIBR is a system command that produces a library listing of files on the user's assigned or a stipulated logical unit (LU). Information relating to each file is displayed in columns under headings as follows:

LIBR Column Heading	Description
*	File type or language - one letter is displayed in this column to identify the file type. See Table 2-2 for a complete list of valid file types.
FILENAME	Complete filename - the password portion is not displayed if the file is a private file on the user's own account or any other account at the same or higher privilege level. A colon is displayed to represent the <CTRL-E>s.
PROT	Files current protection status - see Section 1.4.2 for an explanation of these two digits.
COST	Amount charged by the file's owner each time the file is accessed.
SIZE	Number of disk blocks used to store the file.
ACCOUNT	Group and user number of the user who created the file (in decimal).
AGE	Number of hours since the file was created.
HSLA	Hours since last access - set to zero each time any user accesses the file and increased hourly if the file is not accessed.

Following the complete listing, the number of disk blocks available for creating or expanding files on the selected LU is displayed.

Press <CTRL-S> to halt the display temporarily; press <CTRL-Q> to resume the display. The display may be terminated at any time by pressing <ESC>.

2.18.1 USING LIBR

To start a listing on a user's assigned LU, at the system prompt (#), enter

```
LIBR
```

To list the user's files on another LU, enter

```
LIBR lu/
```

where
lu - number of the logical unit (0-127)

One or more of the following parameters may follow the LIBR command to limit or extend the LIBR listing:

LIBR

Parameter	Description
@	Lists all accessible files on all accounts on the user's assigned LU (default) or a specified LU. An accessible file is any file on the user's own account or on any lower privilege account, or any other file that is not read-protected against this user.
@g	Lists all accessible files that belong to all accounts in group g, where g is a decimal number.
@g,u	Lists only those accessible files that belong to account group g, user u.
*B	Lists only files of type specified by a letter code (see Table 2-2). A maximum of eight types may be specified.
name	Lists only filenames that begin with the characters specified. For example, ACCT would cause listing of filenames such as ACCTA, ACCT3, ACCTXYZ, ACCTEPAS, etc.
<CTRL-E>	Lists only filenames that include a given password; uses the command format **LIBR NAME <CTRL-E>key<CTRL-E>** where key - user's password

^ (caret) — Causes selected entries to be alphabetized before being listed. If the caret is not specified, the files are listed in order of occurrence in the INDEX. An alphabetized listing may be somewhat slower than a nonalphabetized listing.

NOTE

The port's active file is used for alphabetizing the library listing. Therefore, any program in the port's active file will be lost if an alphabetized listing is requested. The port's active file is not disturbed for nonalphabetized library listings.

>h — Lists files that have not been accessed for more than h hours (where h is a decimal number).

<h — Lists files that have been accessed within the last h hours (where h is a decimal number). The >h and <h features may be used together to list only those files last accessed during a certain period of time.

_ (underscore) — Displays only the file type letter and filename columns of the listing. This provides a faster library listing.

[dest{!}] — Outputs the listing to the specified destination, which may be a peripheral device (e.g., $LPT), a text file, or with the exclamation mark (!) to overlay an existing text file.

The parameters listed above may be combined in any sequence. If a logical unit is specified, it must be given first. For example, the command

```
#LIBR 2/TRE *B *T *F @6 ^ >50 <80 [$LPT]
```

prints an alphabetical listing on the line printer of accessible files on LU 2 that meet the following specifications:

- Filename begins with the letters "TRE"
- File is a BASIC program (*B), a text file (*T), or a formatted data file (*F)
- File belongs to any user in account group six (@6)
- File was last accessed more than 50 hours and less than 80 hours ago (>50<80)

2.18.2 EXTENDED LIBR COMMANDS

The information previously given is for general use. Additional information is displayed when the LIBR command is entered from a privilege level 2 account. This information is displayed in three additional columns under headings as follows:

LIBR Column Heading	Description
TYPE	The last three octal digits of the file header's TYPE word. The first digit is the R, L, and I control digit (see Section 2.9.2.1). The other two digits are the file type from which the first column of the library listing is derived. See Table 2-2 for a list of file type codes.
PRIV	The privilege level of the file (same as that of the user who created the file).
HBA	Header block address. The real disk address of the file's header block in octal.

2.18.3 DISK BLOCK USAGE

A level 2 user may obtain a summary of the disk block usage on LU 0 or any specified LU by privilege level. The command form is

```
LIBR ?      (for user's assigned LU)
```

or

```
LIBR lu/?
```

where
lu - number of a logical unit

2.18.4 LIBR AND POLYFILES

A polyfile volume is displayed as an ordinary contiguous (C) file

```
C  filename@xx
```

where
xx - two-digit volume number of a given polyfile

No indication is given as to other volumes that make up a polyfile.

A special program, QUERYPF, is available for listing polyfiles.

2.19 MAIL

MAIL is a system command that permits a user to send a one-line message from one port to another.

To mail a message to another port, at the system command prompt (#), enter

MAIL p message

where
p - number of the destination port

For example, a user on port 7 might send the following message to the manager on port 0:

MAIL 0 -- J. MANAGER, PLEASE INSTALL UNIT 3 <CTRL-G>

The message is received and displayed on port 0 identifying the sender's port number

PORT 7: -- J. MANAGER, PLEASE INSTALL UNIT 3 (bell)

If the destination port is in system command mode, the message is displayed immediately. Pressing <RETURN> redisplays the system command prompt. A message prints even if the destination port is not logged on provided it is switched on.

If a program is running on the destination port, the message will be displayed when that port is awaiting an input.

NOTE

The message may disrupt the screen display on the receiving terminal but does not affect any file. If necessary, use the command appropriate for the program to redisplay the screen.

The sender's port does not perform a return until the message has been received at the destination port.

Two ports may send messages to each other simultaneously. Both messages are sent after each sender has pressed <RETURN>.

To cancel or abort a MAIL command, press <ESC>.

2.20 PORT

The PORT command may be used to change various port characteristics, except where such characteristics are under hardware control. For example, the PORT BAUD command cannot be used to change the baud rate on POINT 4 MARK 2/3 Systems.

PORT EVICT may be used to evict users from their ports as described in Section 2.20.2.

PORT commands do not need a password, except for the PORT EVICT command and those PORT commands that are restricted to or by the system manager. Refer to the IRIS R8 Installation and Configuration Manual for information on setting up passwords and changing restrictions.

To use any of the PORT commands, at the system command prompt (#), enter the command in the form

PORT commandstring

where
commandstring - one of the PORT command forms listed in Table 2-3

If an unauthorized user attempts to use these commands, the message is

ILLEGAL COMMAND OR SYNTAX

Table 2-3 shows the proper command format and explains the function of each command.

TABLE 2-3. PORT COMMANDS

Command Format	Description
PORT ACTIVITY	Displays the number of ports currently in use on the system.
PORT BAUD b	Allows the user to change the baud rate at his terminal (see Section 2.20.1). Cannot be used on POINT 4 MARK 2/3 Systems.
PORT n BAUD b	Changes the baud rate to b on port n. Cannot be used on POINT 4 MARK 2/3 Systems. Restricted to the system manager.

TABLE 2-3. PORT COMMANDS (Cont)

Command Format	Description
PORT DELAY d	Sets the terminal's carriage return delay to the value d, where d is a decimal number not exceeding 127. If the port is on a POINT 4 310 Multiplexer, there will be a delay of d (fiftieth-seconds) after each <RETURN> before the first character of the next line is output. If the port is not on a POINT 4 310 Multiplexer, d null codes are output following a carriage return.
PORT ALL DELAY d	Same as PORT DELAY d but sets the delay for all ports on the system.
PORT DELAY d AFTER c	Sets a delay (as described for PORT DELAY d) to follow the character specified in c. Up to two such special delays may be set, and the same command form may be used to change the delay value. To change the delay character c, first set d to zero.
PORT <CTRL-E>key<CTRL-E>n EVICT	Evicts a user from port n. This command requires a password.
PORT <CTRL-E>key<CTRL-E> ALL EVICT	Evicts all users (except port on which command is executed) at the same time. Requires a password. Restricted to the system manager.
PORT n MONITOR	Displays the account number of the user logged on at port n, the current baud rate of the port, the processor in use, and the program (if any) being run from that port. Restricted to the system manager or other users designated by the system manager.

TABLE 2-3. PORT COMMANDS (Cont)

Command Format	Description
PORT ALL MONITOR	Same as PORT n MONITOR but permits the examination of all ports on the system.
PORT TERMINATOR t	Changes the input termination character to any control character specified in t (cannot be an active character other than <RETURN>). The input terminator is normally a return code. **CAUTION** Many processors require a <RETURN> to terminate input.
PORT n TYPE t	Sets port n to t (where t is the port type of an active terminal translation module). Refer to the IRIS R8 Peripherals Handbook for the correct port type. An invalid port number, inactive module, or an inactive $TERMS system driver results in an error message.
PORT XON	Enables the sending of XON code at the beginning and XOFF codes at the end of each input. Allows controlled input from a device such as a paper tape reader, a magnetic tape cassette, or a floppy disk attached to an interactive port. Sends an octal 21 code (<CTRL-Q> = XON) each time input is enabled if parity is enabled. Sends an octal 223 code (<CTRL-S> = XOFF) immediately after receipt of a terminator code (normally a RETURN, see PORT TERMINATOR). See PORT XOFF command.
PORT XOFF	Disables the transmission of XON and XOFF codes. This is the default condition.

2.20.1 BAUD RATE

The baud rate of a terminal may be changed to any legal baud rate:

110	600	4800
150	1200	9600
300	2400	19200

On MARK 2/3 Systems, the baud rate is set by jumpering the Peripheral Interface Board.

To change the baud rate on the user's own terminal, enter

PORT BAUD b

where
b - any legal baud rate compatible with the system and the user's terminal

After the command has been entered, the speed select switch must be set to the baud rate specified in the BAUD command. If the baud rate was changed successfully, the system command prompt (#) is displayed. Press <ESC>.

Note that some terminals require a delay following a <RETURN>; a longer delay may be required at a higher baud rate. If characters appear to be missing at the beginning of some lines, try increasing the return delay by using the PORT DELAY command.

CAUTION

Do not change your port's baud rate unless the baud rate on your terminal is also changed. The CRT will not be usable but your account will be charged for connect time.

The system manager may change the baud rate of another terminal by using the command

PORT n BAUD b

where
n - port number
b - desired baud rate

Unauthorized use of this command generates the message

ILLEGAL COMMAND OR SYNTAX

If the system manager changes the baud rate while a user is logged on to the port, the following message is displayed:

```
PORT #n ACTIVE CHANGE (Y/N) ?
```

Enter Y to change the baud rate.

If the baud rate is changed successfully, or if N is entered, the system command prompt (#) is displayed at the system manager's terminal.

If the port is inactive (i.e., no user is logged on to that port), a successful change causes the system command prompt (#) to be displayed.

2.20.2 PORT EVICT

The PORT EVICT commands are restricted to the system manager and require the use of a password.

To evict a particular port, at the system command prompt (#), enter

```
PORT <CTRL-E>key<CTRL-E>n EVICT
```

where
key - password assigned to the PORT EVICT command
n - number of the port to be evicted

If all users are to be evicted, at the system command prompt (#), enter

```
PORT <CTRL-E>key<CTRL-E>ALL EVICT
```

where
key - password assigned to the PORT EVICT command

Unauthorized use will result in the message

```
ILLEGAL COMMAND
```

Use of PORT EVICT may cause a port to be locked into the BYE processor if BYE cannot complete its output message. The output message cannot be sent if the port is not configured as a phantom port and the peripheral fails to transmit the log-off message.

2.21 PROTECT

PROTECT is a system command used to scramble BASIC program code so that it cannot be listed even by the owner of the program.

A protected BASIC program can be copied from one logical unit to another. It may be modified by inserting a patch at a specific line number. A protected program cannot be reprotected. If a patch is added, it can be resaved in its protected form. A source listing of the protected program should be kept on a backup disk or tape.

A program may be further protected with a password (key) that is associated with a POINT 4 proprietary device called a Pico-N. The keys that may be used are listed in a Burn Report supplied with each Pico-N. Each BASIC program may have one key only. Use only the key(s) listed in the report to protect a program.

PROTECT should not be confused with file and program protection by which a user may limit access to files. Section 1.4.2 provides information on protection levels and codes.

2.21.1 USING PROTECT WITHOUT A KEY

A program may be protected without the use of the key. The procedure requires two steps. If several programs are to be protected without a key, use U.PROTECT.

1. At the system command prompt (#), enter

   BASIC programname
   <CTRL-C>

   The system command prompt (#) is redisplayed.

2. Enter

   PROTECT <pp>{programname!}

   where

   <pp> - the file protection level given to the program
   programname - the name of the program to be protected
   ! - replaces an existing program file

   Programname is optional at the second step because PROTECT defaults to the programname given in step 1.

2.21.2 USING PROTECT WITH A KEY

Two steps are required to PROTECT a program with a key.

1. At the system command prompt (#), enter

```
BASIC programname
<CTRL-C>
```

   The system prompt (#) is redisplayed.

2. Enter

```
PROTECT <pp>{programname!}, key
```

   where

```
            pp - protection level given to the file
   programname - name of the program
             ! - overlays the existing program
           key - password assigned to PROTECT
```

Programname at the second step is optional because PROTECT defaults to the programname given in the first step.

When the program is protected, it is automatically saved. The PROTECT processor does not detect an invalid key; however, when a key-protected program is run on a system without the proper Pico-N, an error message is displayed

```
PROGRAM IS READ PROTECTED
```

2.22 QUERY

The characteristics of a file or an account status may be determined by use of the QUERY command.

A file's characteristics consist of the following:

- File type (see Table 2-2 for a list of file type codes)
- Protection level
- Size in blocks
- Age
- Last accessed
- Cost
- Total income

Additional information is given for certain types of files; the record length and format of a data file or the starting address of an assembled ASM (stand-alone or executable) program.

If QUERY is initiated from the manager's account, the file's header block address and the R, L, and I control bits are also displayed.

When a user's account status is queried, the following information is displayed:

- Disk blocks available on the user's assigned logical unit (LU)
- Privilege level, account group and number, assigned LU, and priority level
- Number of disk blocks allotted
- Disk blocks in use
- Net accrued charges (i.e., cost of using other files)

2.22.1 QUERYING A FILE'S CHARACTERISTICS

To check the characteristics of a file, at the system command prompt (#), enter

```
QUERY filename
```

where
filename - name of the file to be queried

If the file is not on the user's assigned LU, enter the command in the form

```
QUERY lu/filename
```

where
lu - number of the logical unit

The command is rejected if the file is not found.

If the file is read-protected or copy-protected, only the protection status of the file is displayed. Otherwise, the file's characteristics such as file type, protection, etc., are displayed as shown in the following example:

```
"filename" IS A TEXT FILE

PRIV: 1,  ACCOUNT GROUP: 1,  USER: 3, UNIT: 4
PROTECTION: 77,  SIZE 25 BLOCKS,  AGE: 770 HOURS
LAST ACCESSED: 187 HOURS AGO, COST $0.00 TOTAL INCOME: $0.00

#
```

If the filename is entered incorrectly, or is not on the user's assigned LU, QUERY displays

```
"filename" DOESN'T EXIST
```

If the same file is queried from the manager's account, the following is displayed:

```
"filename" IS A TEXT FILE

HBA  47360,    TYPE: 77030
PRIV: 1,  ACCOUNT GROUP: 1,  USER: 3, UNIT: 4
PROTECTION: 77,  SIZE 25 BLOCKS,  AGE: 770 HOURS
LAST ACCESSED: 187 HOURS AGO, COST $0.00 TOTAL INCOME: $0.00

#
```

Several files may be queried at the same time by entering the command in the form

```
QUERY file1,file2,file3,...
```

Information about each file is then displayed.

2.22.2 QUERYING THE USER ACCOUNT STATUS

To query a user's account status, at the system command prompt (#), enter

```
QUERY @
```

To get account information about any active LU, at the system command prompt (#), enter

```
QUERY @lu/
```

where
lu - number of the logical unit to be queried

The resulting display is similar to the following example:

```
3401 DISC BLOCKS AVAILABLE ON UNIT #4

USER'S ACCOUNT STATUS

PRIV: 1, ACCOUNT GROUP: 1, USER: 3, UNIT: 4, PRIORITY: 5

DISC BLOCKS ALLOTTED: 20000,  DISC BLOCKS IN USE: 17528

COSTS: $0.00 NET ACCRUED CHARGES

#
```

2.22.3 QUERY AND POLYFILES

QUERY may be used on individual volumes of a polyfile. A polyfile volume examined with the QUERY processor appears as a simple contiguous file. It is recommended that QUERYPF be used for polyfiles.

2.23 QUERYPF

QUERYPF is a BASIC program that uses CALL 91 and IRIS Business BASIC SEARCH modes to generate information about the status of existing polyfiles. The information may be directed to the terminal, a file, or a peripheral device such as a line printer. It can be generated in three forms:

- Individual volume display
- Global display
- Informational dump

At the system prompt (#), enter

```
QUERYPF
```

The system responds

```
QUERYPF - Query Polyfiles Utility
Polyfile name [should have LU/]:
```

The polyfile's master volume (volume 0) must be present on the specified logical unit (LU). If no LU is specified, the master volume must reside on the user's assigned LU.

Enter the polyfile with the trailing @ sign.

After the polyfile's master volume is found, QUERYPF prompts

```
Output file [<RETURN> = output to terminal]:
```

Pressing <RETURN> results in output to the terminal. A filename or a device name may be specified for output. After an appropriate response, there is a brief pause while the program gathers the file information. The next prompt is

```
Please input volume number [0-63]
     or <RETURN> for global display
     or -1 for complete dump
     or ESCape to exit to SCOPE:
```

2.23.1 INDIVIDUAL VOLUME DISPLAY

Entry of a volume number causes display of that volume's characteristics as shown in the following example:

```
Volume: 0      DHDR:  1/001130          Logical unit   1 installed.
Privilege level: 2      Group: 1  User 4    Protection: 77
Size: 69 disk blocks
Volume is a Base Directory volume with 3 directories.
Base Volume 0 and its current extensions have a total of 73 blocks for keys
which will hold a maximum of approximately 540 keys.
Directory:  Key length (in characters)
      1: 10     2: 31     3: 25
```

If the volume is not found, an appropriate message is displayed.

2.23.2 POLYFILE GLOBAL DISPLAY

If <RETURN> is pressed, global information for the file is displayed:

```
DEC 28, 1981  12:14:29
Polyfile: 1/DEMOFILE@
Volume 0 LU: 1
            Total data records allocation: 614
Record size is 20 words  Total volumes: 7  Last accessed:  0.09 hours ago

Vol LU/ NBLK Type | Vol LU/ NBLK Type | Vol LU/ NBLK Type | Vol LU/ NBLK Type
 0   1/  69   B 1 |  1   1/  42   D M |  2   2/  11   D M |  3   1/  89   B 4
 4   1/ 100   B 8 |  5   1/  25   E 0 |
```

The display gives volume number, logical unit number, and number of blocks used for that volume in the first three columns. The type column may contain any of the following:

Type	Description
Bnn	Base directory volume whose lowest directory number is nn
Ebb	Directory extension volume whose base volume is bb
D M	Data volume; if the M is present, the volume has a map
U	Unstructured volume; use BUILDPF to structure it
P**	Partially structured file; occurs only when the building of a polyfile was interrupted or aborted
***	Illegal volume type

An asterisk (*) following a volume number indicates a problem with that volume that prevents the polyfile from being opened. Possible reasons for the flag are:

- The HSLA does not match the master volume
- Volume was not found on the LU specified by the master volume
- Volume's LU not installed
- Account number does not match master volume
- Volume is unstructured

2.23.3 COMPLETE DISPLAY

When a -l is entered, a global display followed by individual volume information for each volume in the polyfile is displayed.

```
DEC 28, 1981  12:14:29
Polyfile: 1/DEMOFILE@
Volume 0 LU: 1
            Total data records allocation: 614
Record size is 20 words  Total volumes: 7  Last accessed:  0.09 hours ago
Vol LU/ NBLK Type | Vol LU/ NBLK Type | Vol LU/ NBLK Type | Vol LU/ NBLK Type
 0   1/  69   B 1 |  1   1/  42   D M |  2   2/  11   D M |  3   1/  89   B 4
 4   2/ 100   B 8 |  5   1/  25   E 0 |

Volume: 0      DHDR:  1/014060          Logical unit   1 installed.
Privilege level: 2      Group: 1  User 4    Protection: 77
Size: 69 disk blocks
Volume is a Base Directory volume with 3 directories.
Base Volume 0 and its current extensions have a total of 90 blocks for keys
which will hold a maximum of approximately 540 keys.
Directory: Key length (in characters)
      1: 10     2: 31     3: 25

Volume: 1      DHDR:  1/013661          Logical unit   1 installed.
Privilege level: 2      Group: 1  User 4    Protection: 77
Size: 42 disk blocks
Volume is a Data volume which holds 511 records.
Volume is mapped.  Map size is 1 blocks.

Volume: 2      DHDR:  1/014133          Logical unit   2 installed.
Privilege level: 2      Group: 1  User 4    Protection: 77
Size: 11 disk blocks
Volume is a Data volume which holds 103 records.
Volume is mapped.  Map size is 1 blocks.

Volume: 3      DHDR:  1/014205          Logical unit   1 installed.
Privilege level: 2      Group: 1  User 4    Protection: 77
Size: 89 disk blocks
Volume is a Base Directory volume with 4 directories.
Base Volume 3 and its current extensions have a total of 87 blocks
for keys which will hold a maximum of approximately 1001 keys.
Directory: Key length (in characters)
      4: 17     5: 12     6:  7     7:  5
```

If a logical unit specified in the building process was not installed (in this case LU 2), an asterisk (*) appears next to the volume and the informational display is meaningless. The following is an example of the global display (the individual volume information for volumes 0-3 would be the same as in the previous example):

```
Vol LU/ NBLK Type | Vol LU/ NBLK Type | Vol LU/ NBLK Type | Vol LU/ NBLK Type
 0   1/  69   B 1 |  1   1/  19   E 0 |  2   2/  11   D M |  3   1/  89   B 4
 4*  2/       B 8 |  5   1/  25   E 0 |

Volume: 4      DHDR:  2/??????       ** Logical unit   2 NOT installed. **
Volume is a Base Directory volume with 1 directories.
Base Volume 4 and its current extensions have a total of-2 blocks for keys
which will hold a maximum of approximately 125 keys.
Directory: Key length (in characters)
     44:  0
```

If LU 2 is installed, the display for volume 4 and subsequent volumes is as follows:

```
Volume: 4      DHDR:  2/000043          Logical unit   2 installed.
Privilege level: 2      Group: 1  User 4    Protection: 77
Size: 100 disk blocks
Volume is a Base Directory volume with 1 directories.
Base Volume 4 and its current extensions have a total of 98 blocks
for keys which will hold a maximum of approximately 98 keys.
Directory: Key length (in characters)
      8:120

Volume: 5      DHDR:  1/014336          Logical unit   1 installed.
Privilege level: 2      Group: 1  User 4    Protection: 77
Size: 25 disk blocks
Volume is a Directory Extension of volume 0.
```

When all the volumes have been displayed, the initial prompt is repeated allowing the user to display another polyfile or exit the program:

```
Please input volume number [0-63]
    or <RETURN> for global display
    or      -1 for complete dump
    or  ESCape to exit to SCOPE:
```

2.23.4 QUERYPF ERRORS

QUERYPF attempts to open the file 0/POLYFILERRORS, which contains error messages. If the file is not found, error codes are output and must be looked up by the user. To create 0/POLYFILERRORS, the program BUILDPFERR should be run from the utility account (0,2).

Table 2-4 gives the list of CALL 91 errors. Table 2-5 lists the possible SEARCH mode statuses contained in the file. Both tables give the associated file record numbers.

TABLE 2-4. CALL 91 STATUS CODE

CALL 91 Status	File Rec.#	Description
01	1	Bad channel #
02	2	File not being built
03	3	Bad volume #
04	4	C1 file not polyfile
05	5	Bad filename
06	6	Bad variable type
07	7	Bad number
08	8	Volume preexists on LU; may be part of another polyfile or fragment
09	9	File C0 not found (deleted or polyfilename ≥ 14 characters)
10	10	No nodes
11	11	Volume already defined for this polyfile
12	12	Not used
13	13	Account numbers differ
14	14	Volume in data file table (DFT) but not on disk
15	15	No available volume numbers
16	16	Volume not defined; it is missing
17	17	P does not immediately succeed S in VDT
18	18	P is not an array
19	19	P is not dimensioned at least P[10]
20	20	File C0 is not contiguous
21	21	File C1 is open elsewhere
22	22	File C0 is already a polyfile
23	23	HSLAs do not match for assign operations
24	24	VLU or BNR does not match for assign operations
25	25	Cannot move volume 0
26	26	Illegal move operation
27	27	File C0 is not write-protected
28	28	Illegal write-enable operation

TABLE 2-5. SEARCH MODE STATUS CODE

Status	Rec.#	Description
0	100	No error, operation successful
1	101	Operation not successful
2	102	End of directory
3	103	End of data
4	104	File not indexed
5	105	Polyfile structure error
6	106	Directory number not in sequence
7	107	File is not contiguous
8	108	Volume is already indexed
9	109	Illegal key length (less than 2 or greater than 121)
10	110	Too many directories (limit is 64 per polyfile)
11	111	Volume not found (possible structural error)
12	112	Volume (built) too small
13	113	Directory not found
14	114	File not indexed
15	115	Data volume number is less than the preexisting data volume
16	116	Data volume map request not consistent with preexisting volumes
17	117	Data volume record length does not match that of the polyfile
18	118	Block record out of range
19	119	Record was not allocated (already released)
20	120	Volume has no map

2.24 RUN

A saved BASIC program may be run (executed) directly without calling up the BASIC interpreter. The SMRUN command must be used to run SMbasic programs.

RUN may also be used as a BASIC command. For information on its functions, refer to the IRIS Business BASIC Manual.

A saved BASIC program may be executed using one of the following four command formats:

Command Format	Function
RUN	Causes the system to run the program currently in the port's active file
RUN programname	Causes the system to search only the user's assigned LU for the program
RUN lu/programname	Causes the system to search only the LU specified
programname	Causes the system to search LU 0 first, the default LU second, and then the user's assigned LU.

where
programname - name of saved program
lu - logical unit number

The program is then executed; when it is finished, the system command prompt is displayed.

2.25 SAVE

The SAVE command is used to store on disk a copy of the BASIC program contained in the active file.

Standard IRIS filenaming conventions apply. The program can be stored on any logical unit; optionally, cost and protection can be set.

2.25.1 USING SAVE

To save a program that is in the port's active file, at the system command prompt (#), enter

```
SAVE {<pp> $ddd.cc lu/}filename{!}
```

where

```
      pp - optional protection code
$ddd.cc - optional access cost
      lu - number of another LU
       ! - replaces an old file of the same name
```

If the program has been saved successfully, SAVE displays the message

```
SAVED !!    CHECK CODE = nnn
```

where

nnn - string of characters and numbers

Keep a record of the check codes generated to be used for later verification of the program.

2.25.2 EXAMPLE OF SAVE

An example of a typical SAVE command is

```
SAVE <72> $2.58 4/CASHFLOW3
```

This command saves the object code of the program under the filename CASHFLOW3 on logical unit 4 and protects it against access by lower privilege users. Only the program's creator, other users on the same account, or users with a higher privilege level are able to delete, modify, and resave it under the same name.

Users accessing the program from another account are charged $2.50 (the $.08 is ignored since the cost is carried in 10-cent increments). A record of charges made is accumulated in the program's file header so that the charges can be forwarded to the proper account.

2.25.3 SAVE ERROR MESSAGES

When using SAVE, a number of errors may occur as described in Table 2-6.

TABLE 2-6. SAVE COMMAND ERROR MESSAGES

Error Message	Description
EMPTY FILE	There is no program in the port's active file.
COPY PROTECTED	Program is on another account and is copy-protected.
ILLEGAL FILENAME	The identifier given is not a valid IRIS filename.
NO "!" OR WRONG ACCOUNT	The filename given identifies an existing file with the same name, same type, same logical unit, and on the same user account. Use the form "filename!" (i.e., add an exclamation mark to the filename to replace an existing file). A file belonging to another user cannot be replaced.
FILENAME IN USE FOR DIFFERENT TYPE FILE; NOT SAVED	File is of a different type. Only files of the same type may be replaced by the SAVE command.
FILE BEING CHANGED	File is in the process of being built, replaced, or modified by another user.
LOGICAL UNIT NEEDS n MORE BLOCKS	Logical unit is full, or nearly full, and needs n more blocks to save the program. The system manager should be notified of this condition.
ACCOUNT NEEDS n MORE BLOCKS	Account does not have enough blocks allotted to accommodate this program. If possible, delete some files or programs no longer needed.
INVALID PROTECTION	Protection code specified is invalid.
ILLEGAL COST	Characters other than digits and one period were given following the dollar sign, or a cost greater than $999.90 was specified.

2.26 SMbasic

SMbasic is a programming language supplied as an optional software package. SMbasic can be run only if your system has a specially coded Pico-N.

2.27 SMRUN

On systems with an SMbasic-coded Pico-N, a saved SMbasic program may be run (executed) directly without calling up the SMbasic interpreter.

2.28 U.CHANGE

U.CHANGE is a a BASIC program that creates a jobstream for the CHANGE processor, whereby the protection of a number of files may be changed at the same time. If a single file is to be changed, use the CHANGE command. Use COPYPF to change a polyfile.

U.CHANGE selects files based on parameters entered by the user. Files may not be changed if they are protected against the user.

2.28.1 U.CHANGE WORK FILES

U.CHANGE builds two temporary work files (EDITSV0ppp and EDITSWKAppp) to list files that are to be changed. Files listed in one work file are displayed on the terminal screen for the user to review and select for processing. File(s) remaining on the screen are then written to the second work file.

Each port has exclusive access to these files because the port number is incorporated into the work file names (represented by ppp).

2.28.2 U.CHANGE HELP MODULES

Two help modules are available. The first guides the user in the selection of file parameters. The second help module contains detailed descriptions of the review and selection commands.

To exit either module, press <RETURN>. The program returns to the point where the help module was invoked.

2.28.3 U.CHANGE PROCEDURE

The following is a description of the U.CHANGE procedure.

1. If U.CHANGE is resident on LU 0, the system default LU, or the user's assigned LU, at the system command prompt (#), enter

   U.CHANGE

   If the program resides on another LU, enter the command in the form

   lu/U.CHANGE

   where
   lu - number of the logical unit

   U.CHANGE displays the following:

```
PORT NUMBER nn      FILE CHANGE FACILITY      U.CHANGE 1.1 mm/dd/yy

                    FILE(S) PREFIX: _____________

                    TYPE OF FILE(S): _

                    SOURCE LOGICAL UNIT: __

                    NEW PROTECTION CODE: __


COMMENT:  ENTER A QUESTION MARK (?) AT ANY TIME FOR HELP
COMMAND:
MESSAGE:
```

   The top line of the screen shows the user's port number, program name, program revision number, and release date. File selection prompts are in the center of the display.

   The first help module may be invoked by entering ? in the first position of any field. Explanations of the various options for the parameter entries are then displayed.

   <ESC> may be pressed to abort the current entry and return to the previous field. Pressing <ESC> at the first field aborts the program.

   The three lines at the bottom are for system/user interaction. COMMENT displays system prompts for the user; COMMAND is for user input; MESSAGE displays messages from the system.

2. Enter the file selection parameters. The prompts and the appropriate responses are described below.

   - FILE(S) PREFIX - specify the beginning characters of filenames to be changed. This is particularly helpful when naming conventions have been used to group files into categories.

     Press <RETURN> to list all files except those limited by subsequent parameter entries.

   - TYPE OF FILE(S) - specify the IRIS file type by entering B for BASIC or T for text files, etc. Press <RETURN> to list all files which meet the other parameters.

   - SOURCE LOGICAL UNIT - specify the number of the LU where the files to be changed reside. The default (<RETURN>) assumes the user's own LU.

   - NEW PROTECTION CODE - specify the protection level for those files that are to be changed. There is no default for this field; a response is required. Section 1.4.2 provides a list of valid protection codes.

   After the parameters have been selected with valid or default entries, the program displays

   ARE ALL THE ABOVE SELECTION CRITERIA CORRECT (Y/N) ?

   To return to the selection display, enter N; to begin the file selection process, enter Y. The program displays

   BUILDING WORK FILE, DO NOT DISTURB!

   When initialization is complete, the system displays the specified files in groups of 36. Each group constitutes a page. Files listed on pages that are not reviewed by the user are not processed.

3. To access the second help module for explanations of the review commands, enter 2. Appendix D provides a summary of commands.

   If <ESC> is pressed any time after the first page of filenames is displayed, the program is aborted and the system command prompt (#) is displayed.

4. Use the review commands to remove any files from the screen that are not to be processed.

   Examine all pages of selected files before entering the E (Execute) command.

   The W (Wrap) command allows the user to check files selected for processing. The W command renumbers previously selected files and displays them from the beginning. After the W command is entered, every page must be reviewed.

5. When only the files to be processed remain on the screen (or pages of screens), enter E. A status message is displayed as each file is processed.

6. After the last file is processed, the program asks if the user wants to change more files. To return to the first screen to initiate another session, enter Y; to terminate the program, enter N.

2.29 U.COPY

U.COPY is a BASIC program that creates a jobstream for the COPY processor, whereby selected files are copied from one logical unit to another. If a single file is to be copied, use the COPY command.

U.COPY selects files based on parameters entered by the user.

The U.COPY program does not process the following files:

- Protected against the user
- Driver ($file) files
- DMAP, INDEX, and ACCOUNTS
- Polyfiles

2.29.1 U.COPY WORK FILES

U.COPY builds two temporary work files (EDITSV0ppp and EDITSWKAppp) to list specified files for copying. Files listed in one work file are displayed on the terminal screen for the user to review and modify. File(s) remaining on the screen are then written to the second work file.

Each port has exclusive access to these files because the port number is incorporated into the work file names (represented by ppp).

2.29.2 U.COPY HELP MODULES

Two help modules are available. The first guides the user in the selection of file parameters. The second help module contains detailed descriptions of the review and selection commands.

To exit either module, press <RETURN>. The program returns to the point where the help module was invoked.

2.29.3 U.COPY PROCEDURE

The following is a description of the U.COPY procedure.

1. If U.COPY is resident on LU 0, the system default LU, or the user's assigned LU, at the system command prompt (#), enter

   U.COPY

   If the program resides on another LU, enter the command in the form

   lu/U.COPY

   where
   lu - number of the logical unit

   The U.COPY program displays the following:

```
PORT NUMBER: nn        FILE COPY FACILITY        U.COPY 1.1 mm/dd/yy

                    FILE(S) PREFIX: _____________

                      TYPE OF FILE(S): _

                   SOURCE LOGICAL UNIT: __

                    DEST. LOGICAL UNIT: __


COMMENT:  ENTER A QUESTION MARK (?) AT ANY TIME FOR HELP
COMMAND:
MESSAGE:
```

   The top line of the screen shows the user's port number, program name, program revision number, and release date. File selection prompts are in the center of the display.

   The first help module may be invoked by entering ? in the first position of any field. Explanations of the various options for the requested parameters are then displayed.

   <ESC> may be pressed to abort the current entry and return to the previous field. Pressing <ESC> at the first field aborts the program.

   The three lines at the bottom are for system/user interaction. COMMENT displays system prompts for the user; COMMAND is for user input; MESSAGE displays messages from the system.

2. Enter the file selection parameters as follows:

   - FILE(S) PREFIX - specify the beginning characters of filenames to be copied. This is particularly helpful when naming conventions have been used to group files into categories.

     Press <RETURN> to list all files except those limited by subsequent parameter entries.

   - TYPE OF FILE(S) - specify the IRIS file type by entering B for BASIC or T for text files, etc. Press <RETURN> to list all files that meet the other parameters.

   - SOURCE LOGICAL UNIT - specify the number of the LU where the files reside. The default (<RETURN>) assumes the user's own LU.

   - DEST. LOGICAL UNIT - specify the number of the LU to which the files are to be copied. There is no default for this field; a response is required.

   After the parameters have been selected with valid or default entries, the program displays

   ```
   ARE ALL THE ABOVE SELECTION CRITERIA CORRECT (Y/N) ?
   ```

   To return to the selection display, enter N; to begin the file selection process, enter Y. The program displays

   ```
   BUILDING WORK FILE, DO NOT DISTURB!
   ```

   When initialization is complete, the system displays lists of the specified files in groups of 36. Each group constitutes a page. Files listed on pages that are not reviewed by the user are not processed.

3. To access the second help module for explanation of the review commands, enter ?. Appendix D provides a summary of the review commands.

   If <ESC> is pressed at any time after the first page of filenames is displayed, the program is aborted and the system command prompt (#) is displayed.

4. Use the review commands to remove any files from the screen that are not to be processed.

   Examine all pages of selected files before entering the E (Execute) command.

   The W (Wrap) command allows the user to check files selected for processing. The W command renumbers previously selected files and displays them from the beginning. After the W command is entered, every page must be reviewed.

5. When only the files to be processed remain on the screen (or pages of screens), enter **E**. A status message is displayed as each file is processed.

6. After the last file is processed, the program asks if the user wants to copy more files. To return to the first screen to initiate another session, enter **Y**; to terminate the program, enter **N**.

2.30 U.KILL

U.KILL is a BASIC program that creates a jobstream for the KILL processor, whereby selected files may be deleted at one time. If a single file is to be deleted, use the KILL command.

U.KILL selects files based on parameters entered by the user. The following restrictions apply to the U.KILL program:

- Files protected against the user may not be deleted.
- System files such as DMAP, CONFIG, ACCOUNTS, and INDEX will not appear in the U.KILL work file.
- Processors, Drivers, etc., cannot be deleted unless KILL is used with a password.
- Polyfiles require special handling as described in Section 2.16.3.

2.30.1 U.KILL WORK FILES

U.KILL builds two work files (EDITSV0ppp and EDITSWKAppp) to list specified files for deletion. Files listed in one work file are displayed on the terminal screen for the user to review and modify. The file(s) remaining on the screen are then written to the second work file.

Each port has exclusive access to these files because the port number is incorporated into the work file names (represented by ppp).

2.30.2 U.KILL HELP MODULES

Two help modules are available. The first guides the user in the selection of file parameters. The second help module contains detailed descriptions of the review and selection commands.

To exit either module, press <RETURN>. The program returns to the point where the help module was invoked.

2.30.3 U.KILL PROCEDURE

The following is a description of the U.KILL procedure.

1. If U.KILL is resident on LU 0, the system default LU, or the user's assigned LU, at the system command prompt (#), enter

   U.KILL

   If the program resides on another LU, enter the command in the form

   lu/U.KILL

   where
   lu - number of the logical unit

   U.KILL displays the following:

```
PORT NUMBER: nn     FILE DELETION FACILITY      U.KILL 1.1 mm/dd/yy


                    FILE(S) PREFIX: _____________

                   TYPE OF FILE(S): _

                      LOGICAL UNIT: __


COMMENT:  ENTER A QUESTION MARK (?) AT ANY TIME FOR HELP
COMMAND:
MESSAGE:
```

   The top line of the screen shows the user's port number, program name, program revision number, and release date. File selection prompts are in the center of the display.

   The first help module may be invoked by entering ? in the first position of any field. Explanations of the various options for the parameters are then displayed.

   <ESC> may be pressed to abort the current entry and return to the previous field. Pressing <ESC> at the first field aborts the program.

   The three lines at the bottom are for system/user interaction. COMMENT displays system prompts for the user; COMMAND is for user input; MESSAGE displays messages from the system.

2. Enter the file selection parameters as follows:

   - FILE(S) PREFIX - specify the beginning characters of filenames to be deleted. This is particularly helpful when naming conventions have been used to group files into categories.

     Press <RETURN> to list all files except those limited by subsequent parameter entries.

   - TYPE OF FILE(S) - specify the IRIS file type by entering B for BASIC or T for text files, etc. Press <RETURN> to list all files that meet the other parameters.

   - LOGICAL UNIT - specify the number of the LU where files are to be deleted. The default (<RETURN>) assumes the user's assigned LU.

   After the parameters have been selected with valid or default entries, the program displays

   ```
   ARE ALL THE ABOVE SELECTION CRITERIA CORRECT (Y/N) ?
   ```

   To return to the selection display, enter N; to begin the file review process, enter Y. The program displays

   ```
   BUILDING WORK FILES, DO NOT DISTURB!
   ```

   When initialization is complete, the system displays the specified files in groups of 36. Each group constitutes a page. Files listed on pages that are not reviewed by the user are not processed.

3. To access the second help module for explanations of the review commands, enter ?. Appendix D provides a summary of the review commands.

   If <ESC> is pressed any time after the first page of filenames is displayed, the program is aborted and the system command prompt (#) is displayed.

4. Use the review commands to remove any files from the screen that are not to be processed.

   Examine all pages of selected files before entering the E (Execute) command.

   The W (Wrap) command allows the user to check files selected for processing. The W command renumbers previously selected files and displays them from the beginning. After the W command is entered, every page must be reviewed.

5. When only the files to be processed remain on the screen (or pages of screens), enter E. A status message is displayed as each file is processed.

6. After the last file is processed, the program asks if the user wants to kill more files. To return to the first screen to initiate another session, enter Y; to terminate the program, enter N.

2.31 U.PROTECT

U.PROTECT is a BASIC program that creates a jobstream for the PROTECT processor whereby selected BASIC source code modules are made unlistable. Modules processed by U.PROTECT cannot be listed by any user, regardless of privilege level. Therefore, PROTECT should be used with discretion and a backup copy of the program should be maintained.

Use the PROTECT command if a single program is to be protected.

U.PROTECT selects files based on parameters entered by the user.

The PROTECT procedure should not be confused with file protection codes described in Section 1.4.2, which restrict access to files.

2.31.1 U.PROTECT WORK FILES

U.PROTECT builds two work files (EDITSV0ppp and EDITSWKAppp) to list specified files for protection. Files listed in one work file are displayed on the terminal screen for the user to review and modify. The file(s) remaining on the screen are then written to the second work file.

Each port has exclusive access to these files because the port number is incorporated into the work file names (represented by ppp).

2.31.2 U.PROTECT HELP MODULES

Two help modules are available. The first guides the user in the selection of file parameters. The second help module contains detailed descriptions of the review and selection commands.

To exit either module, press <RETURN>. The program then returns to the point where the help module was invoked.

2.31.3 U.PROTECT PROCEDURE

The following is a description of the U.PROTECT procedure.

1. If U.PROTECT is resident on LU 0, the system default LU, or the user's assigned LU, at the system command prompt (#), enter

   U.PROTECT

   If the program resides on another LU, enter the command in the form

   lu/U.PROTECT

   where
   lu - number of the logical unit

   U.PROTECT displays the following:

```
PORT NUMBER: nn   FILE PROTECT FACILITY    U.PROTECT 1.1 mm/dd/yy

                  FILE(S) PREFIX: _____________

                    LOGICAL UNIT: __


COMMENT:  ENTER A QUESTION MARK (?) AT ANY TIME FOR HELP
COMMAND:
MESSAGE:
```

   The top line of the screen shows the user's port number, program name, program revision number, and the release date. File selection prompts are in the center of the display.

   <ESC> may be pressed to abort the current entry and return to the previous field. Pressing <ESC> at the first field aborts the program.

   The first help module may be accessed by entering ? in the first position of any field. Explanations of the various options for the requested parameters are then displayed.

   The three lines at the bottom are for system/user interaction. COMMENT displays system prompts for the user; COMMAND is for user input; MESSAGE displays messages from the system.

2. Enter the file selection parameters as follows:

   - FILE(S) PREFIX - specify the beginning characters of filenames to be protected. This is particularly helpful when naming conventions have been used to group files into categories.

     Press <RETURN> to list all files except those limited by subsequent parameter entries.

   - LOGICAL UNIT - enter the number of the LU where files are to be protected. The default (<RETURN>) assumes the user's assigned LU.

   After the parameters have been selected with valid or default entries, the program displays

   ```
   ARE ALL THE ABOVE SELECTION CRITERIA CORRECT (Y/N) ?
   ```

   To return to the selection display, enter N; to begin the file selection process, enter Y. The program displays

   ```
   BUILDING WORK FILE, DO NOT DISTURB!
   ```

   When initialization is complete, the system displays the specified files in groups of 36. Each group constitutes a page. Files listed on pages that are not reviewed by the user are not processed.

3. To access the second help module for explanations of the review commands, enter ?. Appendix D provides a summary of the review commands.

   If <ESC> is pressed any time after the first page of filenames is displayed, the program is aborted and the system command prompt (#) is displayed.

4. Use the review commands to remove any files from the screen that are not to be processed.

   Examine all pages of selected files before entering the E (Execute) command.

   The W (Wrap) command allows the user to check files selected for processing. The W command renumbers previously selected files and displays them from the beginning. After the W command is entered, every page must be reviewed.

5. When only the files to be processed remain on the screen (or pages of screens), enter E.

   As each file is processed, the program displays the message

   ```
   PROTECTED  CHECK CODE = nnn
   ```

   where
   nnn - string of characters and numerics

Keep a record of the check codes generated to be used for later verification of the program.

6. When the last file has been protected, the program asks whether the user wants to process more files. To return to the first screen to initiate another session, enter Y; to terminate the program, enter N.

2.32 U.SAVE

U.SAVE is a BASIC program that creates a jobstream for the SAVE processor. Use the SAVE command if a single program is to be saved.

U.SAVE selects files based on parameters entered by the user.

2.32.1 U.SAVE NAMING CONVENTIONS

Program source files (text files) must be identified by a two-character prefix (T.name). U.SAVE truncates the first two characters of a filename, thus distinguishing the source and object code files. For example, a source text file named T.ABCD is SAVEd (and transformed into object code) with the name ABCD.

Files processed by U.SAVE will overwrite existing files of the same account, name, and type.

2.32.2 U.SAVE WORK FILES

U.SAVE builds two temporary work files (EDITSV0ppp and EDITSWKAppp) to list specified files. Files listed in one work file are displayed on the terminal screen for the user to review and select. The file(s) remaining on the screen are then written to the second work file.

Each port has exclusive access to these files because the port number is incorporated into the work file names (represented by ppp).

2.32.3 U.SAVE HELP MODULES

Two help modules are available. The first guides the user in the selection of file parameters. The second help module contains detailed descriptions of the review and selection commands.

To exit either module, press <RETURN>. The program returns to the point where the help module was invoked.

2.32.4 U.SAVE PROCEDURE

The following is a description of the U.SAVE procedure.

1. If U.SAVE is resident on LU 0, the system default LU, or the user's assigned LU, at the system command prompt (#), enter

   U.SAVE

   If the program resides on another LU, enter the command in the form

   lu/U.SAVE

   where
   lu - the number of the logical unit

   U.SAVE displays the following:

```
PORT NUMBER: nn     FILE SAVE FACILITY      U.SAVE 1.1  mm/dd/yy


                  FILE(S) PREFIX: _______________

                    LOGICAL UNIT: __


COMMENT:  ENTER A QUESTION MARK (?) AT ANY TIME FOR HELP
COMMAND:
MESSAGE:
```

   The top line of the screen shows the user's port number, program name, program revision number, and release date. File selection prompts are in the center of the display.

   The first help module may be accessed by entering ? in the first position of any field. Explanations of the various options for the requested parameters are then displayed.

   <ESC> may be pressed to abort the current entry and return to the previous field. Pressing <ESC> at the first field aborts the program.

   The three lines at the bottom are for system/user interaction. COMMENT displays system prompts for the user; COMMAND is for user input; MESSAGE displays messages from the system.

2. Enter the file selection parameters as follows:

   - FILE(S) PREFIX - specify the beginning characters of filenames to be saved. This is particularly helpful when naming conventions have been used to group files into categories.

     Press <RETURN> to list all files except those limited by subsequent parameter entries.

   - LOGICAL UNIT - specify the number of the LU where the files reside. The default (<RETURN>) assumes the user's own LU.

   A list of error messages and explanations may be found in Section 2.25.3 (Table 2-6).

   After the parameters have been selected with valid or default entries, the program displays the message

   ```
   ARE ALL THE ABOVE SELECTION CRITERIA CORRECT (Y/N) ?
   ```

   To return to the selection display, enter N; to begin the file selection process, enter Y. The program displays

   ```
   BUILDING WORK FILE, DO NOT DISTURB!
   ```

   When initialization is complete, the system displays the specified files in groups of 36. Each group constitutes a page. Files listed on pages that are not reviewed by the user are not processed.

3. To access the second help module for explanations of the review commands, enter ?. Appendix D provides a summary of the review commands.

   If <ESC> is pressed any time after the first page of filenames is displayed, the program is aborted and the system command prompt (#) is displayed.

4. Use the review commands to remove any files from the screen that are not to be processed.

   Examine all pages of selected files before entering the E (Execute) command.

   The W (Wrap) command allows the user to check files selected for processing. The W command renumbers previously selected files and displays them from the beginning. After the W command is entered, every page must be reviewed.

5. When only the files to be processed remain on the screen (or pages of screens), enter E.

   As each file is processed, the program displays the message

   ```
   SAVED !!   CHECK CODE = nnn
   ```

   where
   nnn - string consisting of characters and a numeric

   Keep a record of the check codes to be used for later verification of the program.

6. When the last file is saved, the program asks whether the user wants to process more files. To return to the first screen to initiate another session, enter Y; to terminate the program, enter N.

2.33 VERIFY

VERIFY is used to display the check code of a BASIC program. By comparing check codes, a user can ensure that two copies of a program are identical. VERIFY may be used for both IRIS BASIC and SMbasic programs.

Verifying a check code will ascertain whether:

- A program has been modified
- A patch made to a PROTECTed program was done successfully
- An unauthorized modification has been made
- A POINT 4-supplied BASIC program is the correct version
- A program has been transferred successfully from one system to another

To obtain the check code of an IRIS BASIC or an SMbasic program, at the system command prompt (#), enter

```
VERIFY programname
```

To obtain the check code of a program currently in the active file, at the system command prompt (#), enter

```
VERIFY
```

VERIFY will generate an appropriate check code for any file that is accessible to the user. If the file is not accessible, VERIFY may display one of the following messages:

```
? FILE IS COPY PROTECTED, NO CHECK CODE
? FILE IS READ PROTECTED, NO CHECK CODE
```

2.34 XREF

The XREF utility may be used to print a cross-reference dictionary of BASIC program source code.

XREF reads specified text file versions of BASIC programs, then produces the following paginated listings:

- Title page
- Source code list with line numbers referenced by a GOSUB or GOTO statement is prefixed with a plus sign (+)
- Symbol table with BASIC variables in alphabetic sequence with associated program line number references; line number may be tagged to indicate usage
- Channel numbers with line number references followed by BASIC statements with their line number references
- BASIC keywords with line number references
- Program line numbers used as targets arranged in ascending numeric sequence with associated targeting line numbers; targeting line numbers using GOSUB are suffixed with an asterisk (*)
- Cross-reference symbols used by XREF

2.34.1 XREF WORK FILES

XREF builds two temporary work files and one permanent work file to accumulate lists of specified filenames to be processed. Filenames are listed in the temporary file (EDITSV0ppp) and the permanent file (EDITSV0ppp.SAVE). The temporary file (EDITSWKAppp) is used as an input scratch file.

Each port has exclusive access to these files because the port number is incorporated into the work filenames (represented by ppp).

2.34.2 XREF HELP MODULES

Two help modules are available. The first guides the user in the selection of file parameters. The second help module explains the use of the review commands.

To exit either module, press <RETURN>. The program returns to the point where help was invoked.

2.34.3 USING XREF

XREF offers a number of different procedures for file selection and the option of running the program on a phantom port. The user may:

- Create a new work file
- Enter names of programs to be cross-referenced directly into the work file
- Recall the work file (EDITSV0nnn.SAVE) if the procedure was interrupted or other program files need to be cross-referenced

If XREF resides on LU 0, the default LU, or the user's assigned LU, at the system command prompt (#), enter

```
XREF
```

If XREF resides on a different LU, enter the command in the form

```
lu/XREF
```

where
lu - number of the logical unit

The Cross-reference Main Menu is then displayed:

```
PORT NUMBER: nn      CROSS-REFERENCE SELECT        XREF n.n mm/dd/yy



                  FILE(S) PREFIX: ____________

                    LOGICAL UNIT: __

                    LINE PRINTER: _



COMMENT:  CREATE NEW WORK FILE? (Y/N)
COMMAND:
MESSAGE:
```

The top line of the screen shows the user's port number, program revision number, and release date. File selection prompts are in the center of the screen. The three lines at the bottom are for system/user interaction. The COMMENT line displays program prompts, the COMMAND line is for user input, and the MESSAGE line displays system messages.

Pressing <ESC> at a file parameter field aborts the current entry and returns to the previous field. Pressing <ESC> at the first parameter field, at the command line, or while an XREF phase is in progress, aborts the program.

User input must be in uppercase characters.

Respond to the following message displayed on the comment line:

```
CREATE WORK FILE? (Y/N)
```

If new LIBR listing is to be built, enter Y and refer to Section 2.34.3.1.

If a LIBR listing exists, enter N and the program displays

```
COPY FROM SAVED WORK FILE? (Y/N)
```

To use the saved work file, enter Y and refer to Section 2.34.3.3; to terminate the program, enter N.

2.34.3.1 Creating a New LIBR Listing

If a Y was entered at the CREATE NEW WORK FILE prompt, XREF prompts

```
CREATE 'LIBR' WORK FILE? (Y/N)
```

An entry of Y positions the cursor at the first file parameter field. The first help module may be invoked in the first position of any of these fields by entering ?.

Enter the file selection parameters as follows:

- FILE(S) PREFIX - specify the beginning characters of filenames. This is particularly useful where naming conventions have been observed to identify related files or programs by significant prefixes.
- LOGICAL UNIT - enter the logical unit number where the programs reside. The default (<RETURN>) assumes the user's own LU.
- LINE PRINTER - specify the line printer for the XREF output as follows:

```
0 = $LPT (line printer driver to be opened later)
1 = $LPT1
2 = $LPT2
```

After all the parameters have been entered, XREF asks

```
ARE ALL THE ABOVE SELECTION CRITERIA CORRECT? (Y/N)
```

If an error was made, enter N. The entries are erased and the cursor returns to the file selection field.

If everything is correct, enter Y. XREF erases the parameter fields from the screen and displays the following:

```
#LIBR lu/file. *T [<00>EDISV0nn!]

BUILDING WORK FILE, DO NOT DISTURB !
```

When initialization is complete, XREF prompts for the number of copies to be printed

```
ENTER NUMBER OF CROSS-REFERENCE LISTINGS:
```

Enter the number of copies of the listings that are required. The program then prompts

```
ENTER RECIPIENT'S NAME:
```

Enter the name of the person to receive the listings. The program then displays

```
FORMATTING SELECTED FILES. DO NOT DISTURB
```

XREF creates the work file EDITSV0ppp (where ppp is port number). This file is copied to EDITSV0ppp.SAVE for later access if required.

Files are displayed in groups of 36. Each group is a page and each page must be reviewed. Files listed on pages that are not reviewed are not processed. Appendix D provides a summary of the review commands.

Using the review commands, remove any files from the screen that are not to be included in the cross-reference.

The W (Wrap) command is particularly useful for checking the files selected. The W command renumbers the remaining files and wraps to the beginning of the work file. Each page of the revised work file must be reviewed.

The second help module may be invoked at this time. It describes the various edit commands.

When only the files to be processed by XREF remain on the screen (or pages of screens), enter the E command. The message displayed is

```
WRITING SELECTED FILES.  DO NOT DISTURB
```

The next prompt gives the option to run the program on a phantom port as described in Section 2.34.3.4).

When processing is complete, XREF displays

```
PRINTING IN PROGRESS.  DO NOT DISTURB
```

When printing finishes, the program exits and the system command prompt is displayed

```
NORMAL EXIT FROM XREF
#
```

2.34.3.2 Direct Filename Entry

To enter the filenames of programs directly without building a temporary work file, respond as follows:

```
CREATE NEW WORK FILE? (Y/N) Y

CREATE 'LIBR' WORK FILE? (Y/N) N
```

The program prompts for entry of the logical unit number. It then positions the cursor for direct entry of text file names.

After all the filenames have been entered, press <RETURN>. XREF prompts for line printer number and name of recipient. The work file is then displayed. It may be modified with the review commands as described in Appendix D.

2.34.3.3 Reusing Saved Work File

If XREF processing was interrupted for any reason, the saved work file may be reused to continue cross-referencing.

To reuse the saved work file, respond as follows:

```
CREATE NEW WORK FILE? (Y/N) N

COPY FROM SAVED WORK FILE? (Y/N) Y
```

The system requests a line printer number as previously described, then copies a new work file from the saved file while displaying the message

```
PORT #: nn COPY EDITSV0ppp WORK FILE XREFB 2.0  mm/dd/yy

#COPY (00) EDITSV0nn!=EDITSV0ppp.SAVE
```

XREF asks for number of copies required, the recipient's name, and displays the filenames from the old saved file. The list may be modified with the review commands as described in Appendix D.

After the E (Execute) command is given, XREF sends the listings to the printer and returns to the system command prompt (#).

If the files have already been processed, the following screen display will appear:

```
PORT NUMBER: nn   CROSS-REFERENCE SELECT    XREF1 2.0 mm/dd/yy


                  --> NO FILES SELECTED


COMMENT:  'XREF' COMPLETE, MORE PROGRAMS TO PROCESS (Y/N) ?
COMMAND:
MESSAGE:
```

XREF should be terminated and restarted to build a new work file as described in Section 2.34.3.1.

2.34.3.4 Running XREF on a Phantom Port

Upon completion of editing, enter

    E

The program then asks

    RUN ON PHANTOM PORT? (Y/N)

A Y response enables a phantom port and transfers further processing there. The system then prints the message

    RUNNING ON PORT p

where
    p - phantom port number

The program exits and control is returned to system command mode.

If the response is N, the processing of XREF continues on the user's port. The program reports its progress with messages in the center of the screen while the comment line displays the name of the program being processed.

2.34.4 LEGEND OF CROSS-REFERENCE SYMBOLS

The following symbol may appear as a prefix to the statement numbers in the source listing:

Prefix Symbol	Description
+	Statement is the target of one or more GOTOs or GOSUBs

The following symbols may appear as suffixes to the cross-reference line numbers:

Suffix Symbol	Description
:	Variable is declared in a DIM statement Variable is used in a DIM statement to dimension an array or string variable
=	Variable is a target of an assignment
;	Variable is assigned a value as the result of a READ, INPUT, SEARCH, or CALL statement
[	Variable is used as control in a FOR..NEXT (the start of a FOR..NEXT loop)
]	The close of a FOR..NEXT loop
#	Variable is used to designate I/O channel number
*	The line number reference to this statement is a GOSUB

Section 3
IRIS EDITORS

Two editors are supplied with every standard IRIS system: EDIT and FORGE. EDIT is used primarily for text files and Assembly language code, whereas FORGE may be used for IRIS Business BASIC language source code.

Three optional word processing packages, the Electronic Office System (EOS), STYLUS, and TYPIST, are available under IRIS. Each package has its own manual.

3.1 EDIT

EDIT may be used to create or edit IRIS text files.

In the editor, one page is brought into memory and is called the current page. A page consists of a string of characters including <RETURN>s. The end of the page may be delineated by inserting a formfeed character <CTRL-L> as described in Section 3.1.4 and Appendix B.

Page size is limited by the size of the user partition. The size of a user partition is set when the system is configured and may range from 10000 to 77400 octal. If an overflow occurs while loading, inserting, and/or modifying text in the current page, an appropriate error message is displayed. In that case, the COPY command may be used to repaginate the file.

Generally, EDIT commands operate on the current page. Exceptions include commands that append or bring in the next page, commands that search through all pages until a matching string is found, and a command that allows the selection of a new source file.

EDIT displays an asterisk (*) as the EDIT command prompt.

A description of all EDIT command functions is provided in Section 3.1.4. Many of these commands may be combined. An example of such combinations is given in Section 3.1.5.

3.1.1 EDITING AN EXISTING FILE

To edit an existing file, at the system command prompt (#), enter

EDIT sfile,dfile[!]

where

sfile - source file

dfile - destination file

! - indicator that an existing destination file is to be replaced; an optional parameter

Filenames must follow the IRIS naming conventions as described in Section 1.4.

The command may be rejected for one of the following reasons:

- Source file does not exist
- Source file is not a text file
- Source file is read-protected
- Source file is copy-protected
- Destination file exists and no ! was given
- Destination file belongs to another user
- Destination file identifies a file that is not a text file

CAUTION

If an existing file is overlaid by using an exclamation mark (i.e., EDIT sfile,sfile!), data may be lost if the system traps or a <CTRL-C> is pressed.

3.1.2 VIEWING A TEXT FILE

A text file may be viewed without editing it. At the system command prompt (#), enter

```
EDIT filename
```

Any attempt to edit the file will result in an error.

3.1.3 CREATING A NEW TEXT FILE

A new text file may be created with a unique filename. The text can then be entered via the terminal. To create a new file, at the system command prompt (#), enter

```
EDIT .filename
```

The F command may be used to load text from an existing file.

3.1.4 EDIT COMMANDS

EDIT commands are a powerful tool for editing text files and writing program source code. Requirements governing EDIT commands include:

- The following edit commands operate only when a destination file (dfile) is specified:

  ```
  Cstring1/string2
  nCstring1/string2/
  Istring/
  Nstring
  nNstring/
  <CTRL-I>string/
  ```

  If one of these commands is entered and no destination file is specified, the following error message, followed by the EDIT command prompt (*), is displayed:

  ```
  NO DESTINATION FILE!
  *
  ```

- EDIT commands may be entered using either upper or lowercase letters with the exception of XEND and XKIL, which must be uppercase.

- EDIT commands must be followed by a <RETURN>.

- EDIT text strings must be terminated by a string delimiter.

- A forward slash (/) is the default text string delimiter.

- Commands operate only within the current page unless otherwise specified.

- A text pointer is used for most editing functions. The pointer initially precedes the first text character of the current page. After a successful operation, the pointer is positioned at the end of the text that was referenced (except for A, H, 0K, -nK, T, U, V, W and Y commands). The pointer is not displayed. It points between characters and not to a particular character.

- Any pointer move command (J, L, or M) gives no error indication if the beginning or end of the current page is reached. All other commands cause an error message if the command cannot be carried out for any reason.

- Commands may be combined into one command string as described in Section 3.1.5.

Table 3-1 shows the syntax of each EDIT command and provides a description of its functions.

TABLE 3-1. EDIT COMMAND SYNTAX/FUNCTION

Command	Function
A	Appends next page of source file to end of current page. The formfeed character between the two pages is deleted, and the pointer is positioned ahead of the first character of the appended page. If a buffer overflow occurs, an error message is displayed. However, part of the appended page will be retained.
{n}Cstring1/string2/	From pointer, changes n occurrence(s) of string1 in current page to string2. If search is successful, pointer positions after (last) selected string; if not successful, an error message is printed and the pointer is not moved.
nD	Deletes n characters forward from current pointer position.
-nD	Deletes n characters backward from current pointer position.
Estring/	Eliminates all characters from pointer up to, but not including, the first occurrence of the string. Leaves pointer at beginning of the string. If the string is not found on current page, an error message is displayed and the pointer is not moved.
Ffilename	Selects the file identified by filename as the source file to be edited. The current page is first written to the destination file; then, the first page of the newly selected file is brought in as the current page.

TABLE 3-1. EDIT COMMAND SYNTAX/FUNCTION (Cont)

Command	Function
nG	Gets the nth page after the current page of the source file and inserts it into the current page at the current pointer position. Does not change the source page selector. Leaves pointer at beginning of inserted page. If Get causes a buffer overflow, an error message is displayed. However, part of the page will be retained. Cannot be used on an extended text file (over 65535 characters).
0G	Gets a copy of the current page from the source file, inserts it and positions the pointer at the beginning of the page. Cannot be used on an extended file.
-nG,	Same as nG, except gets the nth page preceding the current page.
Hx	Selects any symbol x (not a letter, digit, minus sign, or <RETURN>) as the string delimiter.
Istring/	Inserts string at pointer position. Leaves pointer at end of inserted string. The string may be any string of characters up to but not including the string delimiter. It may include <RETURN>s that are stored as return codes; therefore, the delimiter is required.
<CTRL-I>	Designates a tab within an Insert command.
<CTRL-I>string/	Begins an Insert command and Inserts a tab followed by the string.
{n}J	Jumps (moves pointer) to beginning of nth line on current page. If n is omitted, pointer is moved to the beginning of the first line.

TABLE 3-1. EDIT COMMAND SYNTAX/FUNCTION (Cont)

Command	Function
nK	Deletes (kills) n lines forward from current pointer position.
0K	Deletes from current pointer position back to beginning of line.
-nK	Deletes (kills) n lines backward from current pointer position.
{n}L	Moves pointer to beginning of nth line forward (down) from current position. If n is zero, or is omitted, moves pointer to beginning of current line.
-nL	Moves pointer to beginning of nth line backward (up) from current position.
nM	Moves pointer forward n characters from current position.
-nM	Moves pointer backward n characters from current position.
{n}Nstring/	From pointer, searches for nth occurrence of string. If not found in current page, writes current page into destination file, brings in next page of source file as the current page, and continues searching in this manner. Leaves pointer at end of string. If n is omitted, searches for first occurrence of string.
{n}P	Writes current page into destination file, and brings in nth page of source file as the current page writing intervening pages to destination file. If n is omitted, writes current page and brings in next page from the source file.
0P	Replaces the current page with original from the source file.

TABLE 3-1. EDIT COMMAND SYNTAX/FUNCTION (Cont)

Command	Function
-nP	Moves pointer backward n pages in both source and destination files, deleting current page and any intermediate pages in the destination file. Backs up by counting formfeed characters in both the source and destination files. Cannot be used on an extended text file (over 65535 characters). **NOTE** If page boundaries have been changed by appending pages or by inserting formfeed <CTRL-L> characters, the -nP command should be used only with great caution.
{n}Qstring/	From pointer, searches current and subsequent pages for nth occurrence of string deleting intervening material. Leaves pointer at end of string. If n is omitted, searches current and subsequent pages for first occurrence of the string, deleting intervening material.
nRstream	Repeats the following command stream (up to <RETURN>) n times. Error occurs if any command in stream cannot be repeated n times. Multiline inserts may be repeated by use of <CTRL-Z> in place of <RETURN>. Repeats cannot be nested.
{n}Sstring/	From pointer, searches current page for nth occurrence of string. Leaves pointer at end of string. If n is omitted, searches for first occurrence of the string. If string is not found, the pointer is not moved.

TABLE 3-1. EDIT COMMAND SYNTAX/FUNCTION (Cont)

Command	Function
{n}T	Displays n lines starting at current pointer position. Does not move the pointer. If n is omitted, displays the entire line in which the pointer is positioned.
-nT	Displays from beginning of nth line back from current line through end of current line. Does not move the pointer.
U	Displays number of lines in current page (followed by a semicolon).
V	Displays line number in current page on which pointer is positioned (followed by a colon).
W	Displays number of current page of source file (followed by a period). If pages have been appended, gives the number of the last page appended.
XEND	Exits from editor after writing remainder of source file into destination file (also see <CTRL-C> in Section 3.1.4.1). Must be entered in uppercase.
XKIL	Exits from editor and aborts the destination file. Does not overwrite the source file. Must be entered in uppercase.
Y	Displays number of bytes remaining in the edit buffer (space available for more additions or insertions in the current page).
Z	Moves pointer to end of current page.

3.1.4.1 Special and Control Characters Under EDIT

Special Character	Description
/	Default string delimiter - the slash is recognized as the delimiter unless changed by an H command.
<RETURN>	Command terminator - activates a command and may act as a command string terminator except for the Insert command. Within an Insert command, <RETURN>s may be entered as part of the inserted string; they are stored as return codes and do not terminate the Insert command.
<CTRL-A>	Backspace - generally used with a terminal that does not have backspace capabilities (e.g., Teletype). On most systems, <CTRL-A> causes the character being deleted to be displayed.
<CTRL-C>	Exit command - returns to system command mode. Causes an exit from the editor after writing the current page and closing the destination file.
<CTRL-H>	Backspace - backspaces (if terminal has backspace capability) and deletes the previous character typed. This backspace function may be used repeatedly to delete several characters.
<CTRL-L>	Formfeed - in Insert mode, may be included in a string to force a new page. The resulting code is not displayed; however, a search command may be used to locate the <CTRL-L>. This formfeed code is erased when the A command is used to append a page.
<CTRL-Q>	XON - restarts display on the terminal that had been stopped by a <CTRL-S>.
<CTRL-S>	XOFF - temporarily stops display on a terminal such as the display from a LIBR command.
<CTRL-X>	Cancel input - cancels the entire line just typed in. A backslash symbol (\) is printed to indicate that the line has been deleted, and a carriage return is performed. May be used within an Insert command to delete the current line.
<CTRL-Z>	Return code - where entered as part of a string, it embeds a return code.

3.1.4.2 Special Cases for Using Zero in an EDIT Command

Zero Edit Command	Function
0G	Inserts a copy of the current page from the source file
0K	Deletes from current position back to beginning of line
0P	Replaces current page with its original from the source file

3.1.5 EXAMPLE OF COMBINING EDIT COMMANDS

Individual EDIT commands may be combined to perform a number of functions or to repeat certain functions. The following is an example of combining a number of EDIT commands:

```
*JSSEC7/C7/07/T
```

Figure 3-1 shows the effect on a LIBR listing after the following combination of EDIT commands (which includes a repeat) has been executed:

```
*5KZ-2KJ5RCT  //S /-1DE<CTRL-Z>/1L<RETURN>
```

where

- 5K - kills next 5 lines from current pointer position
- Z - moves pointer to the end of the page
- -2K - kills 2 lines backward from current pointer position
- J - moves pointer to beginning of page
- 5R - repeats the following command string (up to <RETURN>) 5 times
- CT // - changes T and 2 spaces to nothing
- S / - searches for a space
- -1D - deletes 1 character backward from current pointer position (i.e., delete the space)
- E<CTRL-Z>/ - eliminates text from current pointer position to end of line
- 1L - moves pointer to beginning of next line

```
LOGICAL UNIT #10          MAY  9, 1985  15:13:29

*   FILENAME        PROT    COST   SIZE  ACCOUNT  AGE   HSLA  TYPE  PRIV  HBA
T  R82TRMACT5SA      77    $0.00    10    5, 0 18358   1557    30    2    5441
T  R82TRMADDS25SA    77    $0.00    10    5, 0 14932   6360    30    2   13402
T  R82TRMADDS60SA    77    $0.00    18    5, 0 18359   1557    30    2    6445
T  R82TRMADM1SA      77    $0.00    14    5, 0 18358   1557    30    2    7400
T  R82TRMADM2SA      77    $0.00    10    5, 0 16967  15128    30    2   22504

1786 AVAILABLE BLOCKS ON UNIT #10


R82TRMACT5SA
R82TRMADDS25SA
R82TRMADDS60SA
R82TRMADM1SA
R82TRMADM2SA
030-04
```

Figure 3-1. LIBR Listing Before and After a Combined EDIT Command

3.1.6 EDIT COMMAND SUMMARY

The summary of EDIT commands is arranged by the level on which each command functions. Optional parameters are enclosed in braces. For example, a command shown in the format

{{-}n}L

indicates that the minus sign, which gives the direction to move or search backward (i.e., to a previous location), and a specified number (n) are optional and may be used as individual options or combined. In the case of the above example, the command may be used to move the pointer backward or forward to the nth line from the present position. By using the L command without the options, the pointer is moved to the start of the current line.

See Table 3-1 for a definitive description of EDIT commands.

3.1.6.1 Control-level Commands

A control-level command acts on the file as a whole. It may also be used to load another file or parts of another file into the file being edited.

Control-level Command	Function
XEND	Exits EDIT; writes all pages; must be entered in uppercase
<CTRL-C>	Exits EDIT; writes current page only
XKIL	Exits EDIT; aborts destination file; must be entered in uppercase
Ffilename	Finds (selects) the file identified by filename as the source file to be edited
Y	Displays number of bytes remaining in the buffer
nRstream	Repeats the following command stream n times

3.1.6.2 Page-level Commands

Page-level commands function on the current page, or select a preceding or a succeeding page.

Page-level Command	Function
J	Jumps to start of current page
Z	Moves pointer to the end of the current page
U	Displays number of lines in current page
W	Displays current source file page number
{{-}n}P	Pages backward or forward a specified number of pages; writes intermediate pages if paging forward; deletes intermediate pages if paging backward
A	Appends next page of source file to end of current page
{-}nG	Gets the nth page before or after the current page and inserts it at the pointer
{n}Nstring/	Searches for nth occurrence of string on the current or subsequent page; if n is omitted, searches for first occurrence of the string
{n}Qstring/	Searches current and subsequent pages for nth occurrence of a string; deletes unused pages
{n}Sstring/	Searches current page for the nth occurrence of a string

3.1.6.3 Line-level Commands

Line-level commands function on a specified line.

Line-level Command	Function
{n}J	Jumps to the beginning of the nth line on the current page
{{-}n}K	Deletes n+1 lines backward or n lines forward from current pointer position
{{-}n}L	Moves pointer to the beginning of the nth line forward or backward from the current line
V	Displays number of the line on which pointer is positioned
{{-}n}T	Displays n lines from current pointer position; if n is not specified, displays line on which pointer is positioned

3.1.6.4 String-level Commands

String-level commands function on a specified string and position the pointer at the end of the string specified.

String-level Command	Function
{n}Cstring1/string2/	Changes n occurrences of string1 to string2
Estring/	Deletes all characters from the pointer to the beginning of the string
Istring/	Inserts string at the pointer
<CTRL-I>string/	Inserts a tab followed by string at the pointer
Hx	Changes string delimiter to another character (x); the default is / (forward slash)

3.1.6.5 Character-level Commands

Character-level commands function on individual characters on a page.

Character-level Command	Function
{{-}n}D	Deletes n characters forward or backward from the pointer position
{{-}n}M	Moves pointer n characters forward or backward

3.2 FORGE

FORGE is a text editor written in business BASIC under IRIS. It is a useful tool for fast and efficient applications programming, and for modification of text files.

FORGE uses cursor tracking to provide full-screen editing. It has extensive line modification and global search capabilities. Portions of one file may be inserted into another and blocks of code can be saved by creating alternate files. Help modules may be invoked at any time. The user is returned to the point where the edit was suspended.

3.2.1 FORGE WORKFILES

Four workfiles created by the FORGE editor are:

EDITWKAppp

EDITWKBppp

EDITSC0ppp

EDITSV0ppp

where
ppp - number of the user's port

3.2.2 USING FORGE

If FORGE is resident on LU 0, the default LU, or the user's assigned logical unit, at the system command prompt (#), enter

FORGE

If FORGE is resident on another logical unit (LU), enter

lu/FORGE

where
lu - number of the logical unit on which it is resident

FORGE then displays its selection menu as follows:

```
PORT NUMBER: nn          FOR ON-LINE REAL-TIME          FORGE nn MM/DD/YY
                         GENERAL EDITING

                    (0)  EXIT FROM FORGE

                    (1)  EDIT AN EXISTING TEXT FILE

                    (2)  CREATE A NEW TEXT FILE

COMMENT: ENTER THE NUMBER OF THE FUNCTION YOU WISH TO EXECUTE
COMMAND:
MESSAGE:
```

The top of the screen shows the user's port number, the revision number, and release date of the utility. The center of the display lists three options.

Option	Function
0	Returns to the system command prompt (#).
1	Produces a request for a filename and its associated LU. The screen is cleared and the first line of the requested file is displayed. The user can then proceed with editing the file.
2	Produces a request for a new filename and its associated LU. The LU entry may be bypassed by pressing <RETURN> and the system will default to the user's assigned LU. A "last accessed" line is created automatically as the first record of the new file. (This record is updated with the date and time of last access every time the file is modified using FORGE.)

The bottom three lines are for system/user interaction. The COMMENT line provides system prompts for the user, the COMMAND line is for user input, and the MESSAGE line displays messages from the system.

3.2.3 FORGE EDIT COMMANDS

Full-screen editing is a key feature of FORGE. The cursor movement keys or control keys may be used to position the cursor anywhere on the screen allowing the text to be modified directly. For terminals without directional keys, the following control keys may be used:

Keyboard	Control key
<ESC>	<CTRL-D>
Left Arrow	<CTRL-H>
Right Arrow	<CTRL-I>
Down Arrow	<CTRL-J>
Up Arrow	<CTRL-K>

It is possible to perform all the editing functions without using the line commands. Although FORGE is capable of many word processing functions, source code editing is its primary purpose.

FORGE prompts for input by displaying the cursor at the beginning of the bottom line. Each line of new text is scrolled up and the cursor returns to the beginning of the bottom (i.e., next) line.

To request any edit function other than the acceptance of new text, the user must supply that command at the beginning of the bottom line. There is one simple rule: all FORGE commands are preceded by a comma and must start at the beginning of the bottom line.

Whenever a command is typed, the <RETURN> key must be pressed to activate the command. If an error is made at the command line, the faulty entry must be "erased" by entering blanks over the entry.

Entries in excess of 132 characters combined with moving the cursor around a great deal may cause a buffer overflow. To avoid an overflow condition, press <RETURN> periodically.

Table 3-2 shows the format and function of FORGE edit commands.

TABLE 3-2. FORGE EDIT COMMANDS

Command Format	Function
,{nn}	Position of pointer
,A	Abort edit
,C	Copy pointed line
,D{A,B,nn}	Delete line(s)
,E{D,X,#nnn}	End edit
,F 'blank'<string>	Find label
,G[A,B][E,' '][“<filename>“,““]	Group save
,H	Help
,I{nn}	Insert line(s)
,I[“<filename>“>,““] {#nnnnn,/<string>}	Insert file
,L 'blank'<string>	Locate string
,M{nnnnn}' '<string1>[<,/,>]<string2>	Modify line(s)
,N	Show record number
,P	Page forward
,Q	Quit file insert
,R{nnnnn}	Roll through file
,Snnnnn	Skip to record number
,W	Wrap to start of file
,X	Extract and refresh
,@{nn}	Position cursor
,]<string>	Comment pointed line
,]=<character>	Set comment delimiter
,${n}	Print screen
<ESC>	Move pointer up

3.2.3.1 FORGE Command Conventions

Convention	Description
<RETURN>	Commands are entered (i.e., activated) by pressing the return key.
,H	Items not bracketed by adjacent matching delimiters (i.e., [], {}, <>, or single quotes) should be entered exactly as shown. Brackets, braces, and quotes are not entered.
nnn	The lowercase letter n represents a positive numeric value. The number of n's indicates the maximum size of that field.
[A,B]	Items in brackets and separated by commas indicate that at least one of those items must be included in the command.
'blank'	Items in single quotes indicate a single character (i.e., a blank).
{A,B,C}	One of the items, separated by commas and appearing in braces, may be included.
<string>	<> indicates that the item must be replaced by the type of data indicated. Items enclosed in <> and printed in capital letters (e.g., <ESC> <RETURN>) refer to keys on the keyboard.

3.2.3.2 FORGE Error Messages

Number	Description
01	Cursor was moved down past the bottom line
02	General syntax or punctuation error
03	<ESC> was pressed when not allowed
04	Numeric parameter out of valid range
05	In ",M" command, <stringl> not found
06	In ",$" command, driver busy or missing
07	In ",Q" command, not in file insert mode
08	For insert file, already in file insert mode
09	<filename> same as one of edit work files
10	<filename> cannot be accessed properly

3.2.3.3 FORGE Help Module

The FORGE editor has extensive help facilities that may be invoked at any time by entering the command

.H

The first display gives a summary of FORGE commands and a description of error messages. Any individual command may then be entered to invoke a help module for that particular command. A detailed description of the command and its applications is displayed. Some of these modules have more than one page of explanations and instructions.

Further information on a command's syntax conventions may be accessed by entering

?

Press <ESC> to resume editing at the point where help was invoked.

APPENDICES

Appendix A
GLOSSARY

Appendix A contains a glossary of the most commonly used terms under IRIS. Note that when system drivers are enabled, they are prefixed with a dollar sign. The dollar sign has been omitted here (e.g., $DEC is listed in alphabetical sequence as DEC).

Account - an allocation of system time and disk space provided by the system manager for an IRIS user.

Account ID - a string with a maximum of 12 characters assigned to a user and used to log on to the IRIS system.

Account Number - value consisting of a privilege level, group number and user number identifying an IRIS account.

ACCOUNTS - IRIS system file on each logical unit. It contains Account ID, Account numbers, priorities, allotments for the CPU and disk blocks, etc.

ACCOUNTUTILITY - BASIC program used to create and maintain user accounts.

Active file - space on logical unit associated with each interactive port. It is used for swapping and stores a user's current partition between time slices.

ALOAD - BASIC program used to merge binary files.

ANALYPF - BASIC program used to analyze the structure of an existing polyfile.

ASCII - American Standard Code for Information Interchange. This is a 7-bit code used for data transfers in and out of the computer, usually with even parity; i.e., the eighth bit is set to 1 or 0 so that the number of "1" bits in the byte is even. When ASCII is used internally to the IRIS system, the eighth bit is unconditionally set to a 1.

ASM - the IRIS assembly language assembler.

Backup - procedure to save (i.e., copy the system disk) user programs, and data files to another disk, diskette, or tape.

BASIC - the IRIS Business BASIC language interpreter.

BASICTEST - BASIC readiness test.

BCD - binary coded decimal.

Binary tapes - object code paper tapes of the IRIS system files.

Bit - one binary digit, which may be a 1 or a 0.

Block - a unit of disk storage; each block stores 512 8-bit bytes of data.

BLOCKCOPY - stand-alone program for disk-to-disk block transfers.

BTUP - Block Two Utility Package. A low-level debugger, which resides in disk block 2 on logical unit 0.

BUILDPF - BASIC program used to create and extend polyfiles.

BUILDPFERR - BASIC program used to create an error file for the BUILDPF, KILLPF, and QUERYPF programs.

BUILDXF - BASIC program used to create indexed files.

Business BASIC - version of the BASIC language used under IRIS.

BYE - system command used to log off; generates and displays certain acccount information.

Byte - eight bits. One byte may contain an ASCII code or any binary value up to 255.

BZUD - block zero utility driver unique to each disk controller. A copy resides on each logical unit.

CALLTBL - driver containing tables that link BASIC subroutine names and numbers to discsub numbers.

CHANGE - system command used to change file attributes.

CLEANUP - system command used to perform file compression on a logical unit.

CONFIG - system file containing information about the current system configuration.

Contiguous file - IRIS data file that requires allocation of sequential blocks on a disk.

COPY - system command used to copy data from a specified source to a specified destination disk file or peripheral device.

COREMAP - BASIC program used to display current memory allocation.

CPU - computer's central processing unit.

CTR - system software diagnostic routine.

CTUS - physical interface driver for cassette tape units.

Cylinder - a set of tracks on one disk pack that can be accessed without moving the heads. NOTE: A head-per-track disk is considered as having one cylinder (number zero), whereas the cylinder number specifies the head position on a moving arm disk.

DBUG - high-level symbolic debugger.

DDCOPY - disk-to-disk copy utility unique to each disk controller.

DEC - decimal arithmetic system subroutine module.

DEFS - IRIS software definition file.

DFT - data file table for each active port.

DGMX - peripheral driver for a Data General 4060-type multiplexer.

Dirty page - buffer containing data that has not been written to disk.

DISCSUBS - file on logical unit number zero containing IRIS system subroutines.

DISCSUBS.USER - file used for OEM-supplied discsubs.

Disk address - identifies a particular disk block on a given logical unit. See also RDA, logical address, and physical address.

Disk address list - contains RDAs on the logical unit.

Disk partition - subdivision of a physical disk where logical units are installed.

DMAP - disk map for each logical unit; indicates which blocks are in use and which are available.

DOOM - bits set to enable program access to certain restricted system functions.

DSP - disk service processor, a symbolic machine code editor.

EDIT - IRIS text file editor.

EXTRAPORT - system command used to run programs on a phantom port.

File header - first block of a file; contains information about the file.

Filename - unique, user-supplied name for a data file or application program.

FOREIGN - peripheral driver allows read/write access to or from any kind of disk regardless of file system.

Foreign LU - logical unit number dedicated to read/write operations for "foreign" disks.

FORGE - editor for text files.

FORMAT - system command used to create formatted or contiguous files.

Formatted files - IRIS data file consisting of user-defined fields (items) within each record.

Full configuration - a complete IPL activating the whole system.

GUARD - Program used to manipulate DOOM bits.

GUIDE - set of BASIC programs (GUIDE.LU, GUIDE.LPT, GUIDE.BLKCOPY) that give directions for the configuration of a logical unit, line printer drivers, and the use of BLOCKCOPY.

INDEX - file directory containing filenames and RDAs on each logical unit.

Indexed files - IRIS data file with directories containing self-sorting keys for keyword file access.

INFO table - system information table in the CONFIG file containing system parameters.

INSTALL - system command used to install a logical unit.

IPL - initial program load; brings the IRIS Operating system into memory from disk.

IRIS - Interactive Real-Time Information System.

KILL - system command used to delete files.

KILLPF - system command used to delete polyfiles.

Latency time - time required for a disk to revolve so that a selected sector is positioned under the read/write heads.

LIBR - system command used to list files.

Logical address - address of a logical cylinder, logical track, and logical sector. It is generally used only by the allocate and deallocate subroutines and some system drivers.

Logical cylinder - usually equal to the physical cylinder minus the starting cylinder of the logical unit. If the physical unit is partitioned into two or more logical units, the physical cylinder number is derived by adding the logical cylinder to the number of the first real cylinder.

Logical sector - a function of a physical sector (see also sector).

Logical track - a function of a physical track (see also track).

Logical unit - a partition of a physical disk unit.

LPTD - universal driver for a line printer on a Data General 4060-type multiplexer port.

LPTM - universal driver for a line printer on a POINT 4 310 or MARK 3 multiplexer port.

LPTP - universal programmed I/O line printer driver (device code 17).

LU - logical unit.

LUFIX - logical unit fixed information table.

LUVAR - logical unit variable information table (specifies the physical disk location).

MAIL - system command used to send a message from one port to another.

MESSAGES - IRIS file containing standard error messages.

Minimum configuration - an IPL procedure which activates the master terminal only.

MMUX - POINT 4 multiplexer driver.

MTA0 - interface driver for a magnetic or cassette tape unit.

MTAS - physical interface handler for magnetic tape units.

Mux - multiplexer.

MUX310DP - POINT 4 310 Multiplexer diagnostic.

Open file maintenance - procedure for making restricted system files accessible from selected BASIC programs.

Partition - one or more equal areas of memory allocated to a user for the execution of programs (see also disk partition).

Passive file - any file which has been stored (i.e., saved).

PCB - port control block.

PCW - port control word.

PDT - port definition table.

PHA - phantom port driver.

Phantom port - port without a terminal; used to run jobs that do not require operator interaction.

Physical address - disk address used by the disk controller.

Pico-N - POINT 4 encoded proprietary device; prevents unauthorized use of IRIS, application packages, etc.

Polyfile - IRIS data file that can accommodate a large data base and has keyword access.

Port - interactive I/O channel on the IRIS system.

PORT - system command used to change port attributes or display port activity.

Private file - read/write-protected file with a password to prevent unauthorized access.

Processor - system program that is activated by a command word.

PROTECT - system command used to scramble a program's object code, making it unlistable.

PSIZ - Partition size (i.e., maximum BASIC Program size) parameters in the CONFIG file's General Information Table.

PTM - driver for a master Teletype reader/punch.

PTP - driver for a paper tape punch.

PTR - driver for a paper tape reader.

PZ - page zero definitions file.

QUERY - system command used to display file characteristics.

QUERYPF - system command used to display polyfile characteristics.

RDA - real disk address; block number relative to the start of a logical unit.

Real address - See RDA.

Recursive - procedure that invokes itself internally and must be re-entrant.

Re-entrant - routine, usually a subroutine, that may be simultaneously in use by several users since it is not modified during execution.

REHASH - system command used to reposition file entries in a logical unit INDEX for faster access.

REMOVE - system command used to close a logical unit.

REX - real-time executive, contains system-level modules.

Routine - any sequence of instructions or program statements.

RUN - run-time processor used to execute a BASIC program.

RUNMAT - processor used to execute BASIC language matrix algebra.

SAVE - system command used to save a BASIC program to disk.

SCO - software change order.

SCOPE - system command processor that analyzes all system commands, calls a selected program, and displays the system command prompt (#).

Sector - fraction of a revolution of a disk. It may be thought of as a slice of pie. One block of data may be recorded on each track within this "slice".

Seek time - time required for the read/write heads to move to the cylinder on disk where the desired data is stored.

SETTIME - BASIC program used to set the system date and time.

SETUP - IRIS system configurator.

SHUTDOWN - system command used to shut down the system.

SIR - system initializing routine; performs IPL functions.

SMBASIC - system command used to invoke the SMbasic interpreter.

String - one or more bytes (usually ASCII codes) terminated by a byte consisting of all zeros.

Subroutine - self-contained program module that can be accessed from another routine.

Surface - one side of one platter of a disk or disk cartridge.

SYMBOLS - file containing assembler symbols.

SYSBAK - on-line utility program for performing disk-to-tape or disk-to-floppy backups.

Sysgen - system generation procedure via paper tape.

TERM.name - terminal translation module unique to each type of terminal.

TERMS - terminal translation system subroutine module.

Text file - IRIS data file consisting of a single string of zero to 16,777,215 characters.

Track - path traced on a disk surface by one head in one position.

Trap - system error condition caused by a program or hardware error.

TTY - driver for a Teletype or CRT.

VERIFY - system command used to retrieve the checksum associated with SAVEd BASIC programs.

Word - 16 bits (2 bytes).

XREF - system command used to cross-reference a BASIC program.

Appendix B
TERMINAL COMMAND KEYS

Several keys on the terminal keyboard are significant to the IRIS Operating System. Most require that the control key (<CTRL on the keyboard>) be held down while another key is pressed; both are then released simultaneously. For example, <CTRL-A> requires holding down the <CTRL> key and pressing the A key.

Characters typed by the user are not examined by the processor until the <RETURN> key is pressed. Typing errors may be corrected before being detected by the processor, thereby saving time and preventing error messages, or it may be desirable to stop or escape an action. It is also possible to imbed characters in a string to cause a certain action. Table B-1 gives a description of these terminal commands. Table B-2 provides the ASCII code chart.

TABLE B-1. TERMINAL COMMAND KEYS

Command	Description
<CTRL-A>	Backspace - generally used with a terminal such as a Teletype that does not have backspace capabilities. On most systems, <CTRL-A> causes the character being deleted to be printed instead of echoing a backspace code.
<CTRL-B>	Break - causes a signal to be sent to the user's own port. Both signal values are zero. May be used in a program to recognize a user's request while the program is in a processing loop.
<CTRL-C>	System escape - permits the user to return to system control or command mode.
<CTRL-D>	Processor escape - an alternative to the <Processor ESC> key.

TABLE B-1. TERMINAL COMMAND KEYS (Cont)

Command	Description
<CTRL-E>	Echo toggle - characters typed in following the <CTRL-E> will not be echoed. Pressing <CTRL-E> a second time restores the echo.
<CTRL-G>	Bell - may be included in a printed string to ring the terminal's bell. It is entered into the input buffer.
<CTRL-H>	Backspace - deletes the previous character typed. The backspace function may be used repeatedly to delete several characters. The backspace code itself (<CTRL-H>) is echoed; this causes a terminal having backspace capabilities to backspace its cursor so that the new character may be displayed at the same position as the character that was deleted.
<CTRL-I>	Horizontal tab - may be included in a string to cause the line printer or terminal to advance the printing position to the next tab stop. A single space is displayed on terminals lacking tab stop capability.
<CTRL-K>	Vertical tab - may be included in a printed string to cause a line printer or some terminals to advance to the next vertical tab stop.
<CTRL-L>	Formfeed - may be included in a printed string to cause the line printer or some terminals to advance to the top of the next page.
<CTRL-M>	Same as <RETURN>.
<CTRL-O>	Stop output - causes the termination of the current display but allows the next display to be started. May be used to stop the display of known messages.
<CTRL-P>	Toggle parity - toggles parity checking. IRIS either checks for parity or ignores parity. Checking is the default. A percent sign (%) is echoed.
<CTRL-Q>	XON - restarts output to the terminal that had been paused by a <CTRL-S>.

TABLE B-1. TERMINAL COMMAND KEYS (Cont)

Command	Description
<CTRL-S>	XOFF - pauses output to a terminal such as the display resulting from a LIBR command.
<CTRL-X>	Cancel input - cancels the entire line just typed in. A backslash (\) is printed to indicate that the line has been deleted, and a carriage return is performed. Input mode remains enabled.
<CTRL-Y>	Bypass error branching - if a BASIC program is running with error branching in effect, <ESC> and <CTRL-C> will not work. They would cause a trappable error (Error 99) instead of terminating the procedure. However, if a program is <u>not</u> copy-protected against the user, a <CTRL-Y> preceding an <ESC> terminates the procedure in the normal manner.
<CTRL-Z>	Insert carriage return - when using several IRIS processors, embeds a return code as part of a string.
<CTRL-_>	Bell - causes a BEL code (i.e., a bell rings) to be echoed. Differs from <CTRL-G> because nothing is entered into the input buffer.
<ESC>	Terminator - terminates any current or pending output and/or procedure. It terminates <CTRL-S> pause mode, prints a \, performs a carriage return, and returns the system to command mode. It may be used to escape from an undesired or unknown situation. See <CTRL-Y> for situations where <ESC> does not terminate a process. <ESC> is also used to activate the terminal when signing on.
<ALT MODE>	Same as <ESC> on some systems.
<RETURN>	End input - each command or command string (unless otherwise noted) must be entered into the system by <RETURN> (carriage return). If enabled by a PORT XON command, an XOFF code is echoed to stop the tape reader.
<BREAK>	Same as <CTRL-B>.

TABLE B-1. TERMINAL COMMAND KEYS (Cont)

Command	Description
<HERE IS>	Null - on Teletype-type terminals, the HERE IS-key transmits 20 null codes (all zero bytes). Nulls are ignored by IRIS but they are echoed. May be used to punch leader on the terminal's paper tape punch. If the HERE IS-key on your terminal does not generate nulls, you may use <CTRL-SHIFT-REPT-P> keys simultaneously. The REPT and P keys must be released first to avoid generating garbage codes.

In some cases, a typed character may not be accepted by the computer. Such cases are:

- Input not enabled
- Transmission error
- Input buffer full
- Parity is incompatible

If input is not enabled, the character (other than <ESC> or <CTRL-P>) is retained in an intermediate input buffer (IIB) until input is enabled, at which time it is echoed and entered into the I/O buffer.

In the case of a transmission error, parity error, or buffer-full condition, the terminal's bell rings. Try typing the same character again.

- If it is a transmission error, the character may be accepted the second time; if not, press <CTRL-P> to suppress parity checking (a % symbol is printed) and try again.
- If the input buffer is full, one of the following must be used before further input will be accepted:

  <CTRL-A>

  <CTRL-H>

  <CTRL-X>

  <ESC>

TABLE B-2. ASCII CODE IN OCTAL

000	NUL	<CTRL-@>	040	BLANK	100	@	140	`
001	SOH	<CTRL-A>	041	!	101	A	141	a
002	STX	<CTRL-B>	042	"	102	B	142	b
003	ETX	<CTRL-C>	043	#	103	C	143	c
004	EOT	<CTRL-D>	044	$	104	D	144	d
005	ENQ	<CTRL-E>	045	%	105	E	145	e
006	ACK	<CTRL-F>	046	&	106	F	146	f
007	BEL	<CTRL-G>	047	'	107	G	147	g
010	BKSP	<CTRL-H>	050	(	110	H	150	h
011	HTAB	<CTRL-I>	051	)	111	I	151	i
012	LF	<CTRL-J>	052	*	112	J	152	j
013	VTAB	<CTRL-K>	053	+	113	K	153	k
014	FF	<CTRL-L>	054	,	114	L	154	l
015	CR	<CTRL-M>	055	-	115	M	155	m
016	SO	<CTRL-N>	056	.	116	N	156	n
017	SI	<CTRL-O>	057	/	117	O	157	o
020	DLE	<CTRL-P>	060	0	120	P	160	p
021	XON	<CTRL-Q>	061	1	121	Q	161	q
022	AUXON	<CTRL-R>	062	2	122	R	162	r
023	XOFF	<CTRL-S>	063	3	123	S	163	s
024	AUXOFF	<CTRL-T>	064	4	124	T	164	t
025	NAK	<CTRL-U>	065	5	125	U	165	u
026	SYN	<CTRL-V>	066	6	126	V	166	v
027	ETB	<CTRL-W>	067	7	127	W	167	w
030	CAN	<CTRL-X>	070	8	130	X	170	x
031	ENDMD	<CTRL-Y>	071	9	131	Y	171	y
032	SUB	<CTRL-Z>	072	:	132	Z	172	z
033	ESC	<CTRL-[>	073	;	133	[	173	{
034	F SEP	<CTRL-\>	074	<	134	\	174	|
035	G SEP	<CTRL-]>	075	=	135	]	175	}
036	R SEP	<CTRL-^>	076	>	136	^	176	~
037	U SEP	<CTRL-_>	077	?	137	_	177	DEL

NOTE

The IRIS system stores all ASCII codes with the most significant bit set. Add 200 octal to each of the above values to obtain IRIS ASCII codes. For example, SOH has an octal value of 201.

Appendix C
BUILDPF EXERCISES

The BUILDPF exercises are intended to acquaint the user with the process of creating and extending a polyfile. The first exercise gives a series of specifications for the creation of a polyfile followed by an exercise. The second exercise specifies the extension of the file and includes that portion of the exercise.

Please familiarize yourself with the description of the BUILDPF command given in Section 2.6.

Ask the system manager which logical units (LU) are available. LU 1 and LU 2 are used in this exercise; you may wish to use different LU numbers. If a logical unit is not installed or the user does not have an account assigned to it (i.e., cannot access the LU), an Error 26 (logical unit not active) occurs and the program aborts.

User input is underlined and requires a <RETURN> unless otherwise noted. <RETURN> is not shown unless it is the only response required.

C.1 CREATING A POLYFILE

The usual practice for planning a new file is to set up specifications for it by giving consideration to the ultimate number of data records, the number and size of keys required to give adequate access to the data, and making sure that the filename is unique but still reflects the purpose of the file in some way. Check that the LU(s) on which the volumes are to reside have enough contiguous blocks to accommodate the file. When there is not enough space, an Error 80 is returned and the BUILDPF program aborted. If that happens, running CLEANUP (see the IRIS R8 Operations Manual) may create enough contiguous space.

In practice, it is strongly recommended that the user let the system assign volume and directory numbers for greatest efficiency. Although the program displays a message if a volume is already in use and QUERYPF can be run to find out what volume numbers have been assigned, this is time-consuming. Remember that data volumes can be added only to existing data volumes; they cannot be inserted. If a polyfile has data volumes numbered 4, 5, and 10, a new data volume numbered 8 cannot be built because record numbers are assigned sequentially through all data volumes; they are not renumbered when a new data volume is built.

When building a number of directory volumes, BUILDPF first asks for the key size for each directory. It then asks for "volume size in indexes (keys)". This refers to the <u>total</u> number of keys for all the directories in a particular volume.

C.1.1 SPECIFICATIONS FOR BUILDING A NEW POLYFILE

For the purpose of this exercise, the specifications are arbitrary and are intended to demonstrate the use of BUILDPF.

1. Polyfile name is DEMOFILE@.

2. Master volume (volume 0) is to reside on LU 1.

3. Three other directory volumes are required. Start these with volume #7 to leave room for possible future extensions.

   - Directory 1 with 100 keys each 10 bytes long
   - Directory 2 with 100 keys each 31 bytes long
   - Directory 3 with 200 keys each 25 bytes long

   In practice, a base directory volume can have multiple directories and the blocks contained in the extension volume can be used for all the directories in the base directory volume.

4. One data volume is required on LU number 1 containing 500 records of 20 words each and it is to be mapped. Let the system assign the volume number.

   In practice, it is always better to have mapped data volumes because SEARCH mode 1, dir=0 (see the IRIS Business BASIC Manual) can get and return free records. If data volumes are not mapped, the application program must keep track of blocks in use.

C.1.2 EXERCISE FOR CREATING A NEW POLYFILE

Whether a new polyfile is to be created or an old one extended, the initial command is the same. At the system command prompt (#), enter

```
BUILDPF
```

BUILDPF responds

```
BUILDPF - Build Polyfile Utility
```

The response shown below will give the results specified in Section C.1.1:

```
Polyfilename [must have "LU/" (not 0)]: 1/DEMOFILE@
Polyfile not found.  Do you wish to create one? Y
Creating a NEW polyfile.
Record size (in words) for the entire Polyfile: 20
Volume: 0
```

NOTE

The master volume (volume 0) is built first by the BUILDPF program automatically.

```
Volume types:  "B"  Base Directory
               "E"  Extension Directory
               "D"  Data

Volume type: B

Starting directory number for this volume: 1
Directory numbers available from 1 thru 63
Directory 1  Keys size in characters [<RETURN> to terminate]: 10
Directory 2  Keys size in characters [<RETURN> to terminate]: 31
Directory 3  Keys size in characters [<RETURN> to terminate]: 25
Directory 4  Keys size in characters [<RETURN> to terminate]: <RETURN>

This directory setup ok? [<RETURN> = ok]: <RETURN>
```

The response to the last prompt is optionally <RETURN> or Y if the setup is all right.

NOTE

BUILDPF program automatically assigns sequential directory volume numbers once the first directory volume number has been entered.

The program then prompts for the total number of keys desired and proceeds to structure the volumes.

```
Volume size in indexes (keys)
  [Negative for size in blocks]: 400
Allocating volume.  Please wait.
Volume 0 allocation complete.
Structuring volume 0 as base directory volume.  Please wait.
Directory:  1  2  3  Structuring complete.
```

With the following prompts, we have the option of exiting the program by pressing <RETURN>. However, we still have to build the data volume. To do this, respond to the prompts as follows:

```
Extend Base Volume 0 more ["Y"/"N"; <RETURN> = Exit]:  N

Logical Unit (nonzero) for volume [<RETURN> = Exit]: 1

Volume number [ 0-63; <RETURN> = don't care]: <RETURN>

Volume types: "B"  Base Directory
              "E"  Extension Directory
              "D"  Data

Volume type: D

Data Volume(s) to have maps? ["Y"/"N"]: Y

Volume size in records
  [Negative for size in blocks]: 500
Allocating volume.  Please wait.
Volume 1 allocation complete.
Structuring volume 1 as data volume.  Please wait.
Structuring complete.
Logical Unit (nonzero) for volume [<RETURN> = Exit]: <RETURN>
#
```

When a <RETURN> is entered in response to the logical unit prompt, BUILDF returns the user to the system command prompt and the creation of a polyfile is completed.

C.2 EXTENDING A POLYFILE

After a period of time, it may be necessary to add another data volume and other directories to a polyfile. That procedure is very similar to creating a file.

It is advisable to plan ahead by setting up some specifications for the additional volumes. However, the record length remains the same and, because the first data volume was mapped, any additional data volumes must be mapped also.

To find out exactly what the attributes of the existing polyfile are, run QUERYPF. Make a note of the blocks used, volume numbers that were assigned, on which LUs the volumes reside, etc. Then, make an outline of the extension requirements.

C.2.1 SPECIFICATIONS FOR EXTENDING A POLYFILE

The specifications given for this exercise are arbitrary. They are intended to demonstrate polyfile features and are not a guideline for a particular file configuration.

1. Create one more data volume with the next available volume number using 10 blocks.

2. Create two base directories; let the system assign the volume numbers.

   a. One base directory on LU 1 with four directories, giving a total of 1000 keys.

      - One directory with 200 keys, each 17 bytes long
      - One directory with 200 keys, each 12 bytes long
      - One directory with 300 keys, each 7 bytes long
      - One directory with 300 keys, each 5 bytes long

   b. One base directory volume on LU 2 (directory 8) with a keylength of 120 bytes.

3. Add a directory extension on LU 2 to volume 0 to add 25 blocks to the base directory. Make it number 5.

C.2.2 EXTENDING POLYFILE EXERCISE

In this exercise, the responses given for the prompts correspond to the specifications given in Section C.2.1. To extend a polyfile, at the system command prompt (#), enter

```
BUILDPF

BUILDPF - Build Polyfile Utility

Polyfilename [must have "LU/" (not 0)]: 1/DEMOFILE@
File found.  Polyfile EXTENSION mode.
Logical Unit (nonzero) for volume [<RETURN> = Exit]: 2
Volume number [ 0-63; <RETURN> = don't care]: <RETURN>
Volume types: "B"  Base Directory
              "E"  Extension Directory
              "D"  Data

Volume type: D

Volume size in records
  [Negative for size in blocks]: -10
Allocating volume.  Please wait.
Volume 2 allocation complete.
Structuring volume 2 as Data Volume.  Please wait.
Structuring complete.

Logical Unit (nonzero) for volume [<RETURN> = exit]: 1
Volume number [ 0-63; <RETURN> = don't care]: <RETURN>
Volume types:  "B"  Base Directory
               "E"  Extension Directory
               "D"  Data

Volume type: B
Starting directory number for this volume: 4
Directory numbers available from 4 thru 63
Directory 4  Keys size in characters [<RETURN> to terminate]: 17
Directory 5  Keys size in characters [<RETURN> to terminate]: 12
Directory 6  Keys size in characters [<RETURN> to terminate]: 7
Directory 7  Keys size in characters [<RETURN> to terminate]: 5
Directory 8  Keys size in characters [<RETURN> to terminate]: <RETURN>

This directory setup ok? [<RETURN> = ok]: Y
Volume size in indexes (keys)
  [Negative for size in blocks]: 1000
Allocating volume.  Please wait.
Volume 3 allocation complete.
Structuring volume 3 as a directory extension.  Please wait.
Directory:  4  5  6  7  Structuring complete.
Extend Base Volume 0 more ["Y"/"N"; <RETURN> = exit]:  N
Logical Unit (nonzero) for volume [<RETURN> = Exit]: 2
Volume number [ 0-63; <RETURN> = don't care]: <RETURN>
Volume types: "B"  Base Directory
              "E"  Extension Directory
              "D"  Data
```

```
Volume type: B
Starting directory number for this volume: 8
Directory numbers available from 8 thru 63
Directory 8  Keys size in characters [<RETURN> to terminate]: C
Directory 8  Keys size in characters [<RETURN> to terminate]: 120
```

If an alpha character is entered instead of a numeric value, the prompt for a given directory volume is redisplayed. If a wrong value is entered, the user can cancel the entry by pressing <RETURN> at the next directory prompt and entering N at the prompt.

```
This directory setup ok? [<RETURN> = ok]: N
```

The program then returns to the beginning at the directory setup procedure.

```
Starting directory number for this volume: 8
Directory numbers available from 8 thru 63
Directory 8  Keys size in characters [<RETURN> to terminate]: 120
Directory 9  Keys size in characters [<RETURN> to terminate]: <RETURN>
This directory setup ok? [<RETURN> = ok]: Y
Volume size in indexes (keys)
  [Negative for size in blocks]: 100
Allocating volume.  Please wait.
Volume 4 allocation complete.
Structuring volume 4 as Base Directory Volume.  Please wait.
Directory:  8  Structuring complete.
Extend Base Volume more ["Y"/"N"; <RETURN> = exit]: N
Logical Unit (nonzero) for volume [<RETURN> = Exit]: 1
Volume number [ 0-63; <RETURN> = don't care]: <RETURN>
Volume types:  "B"  Base Directory
               "E"  Extension Directory
               "D"  Data

Volume type: E
Volume to extend: 0
Base Volume 0 and its current extensions have a
total of 67 blocks which will hold a maximum of
approximately 402 keys.
New maximum number of indexes (keys)
  [Negative for size in blocks]: -25
Allocating volume.  Please wait.
Volume 5 allocation complete.
Structuring volume 5 as a directory extension.  Please wait.
Structuring complete.
Extend Base Volume 0 more ["Y"/"N"; <RETURN> = exit]: N
Logical Unit (nonzero) for volume [<RETURN> = Exit]: <RETURN>
#
```

As a final step, we should make sure that the file was built according to the specifications. Run QUERYPF to give a global display of the file. The display should be similar to the one shown in Section 2.23. Then, use KILLPF to delete the polyfile.

Appendix D
U.UTILITY REVIEW COMMAND SUMMARY

The various U.Utilities perform their functions on files selected by the user. After the selection parameters have been entered, the user may make further selections using the U.Utility review commands described below.

Command	Function
A	All - erases all file names on the current screen. None of the files erased will be copied.
E	Execute - processes those filenames that are left on the screen after the last page has been reviewed and each file is processed in turn.
P	Page - stores for later processing the filenames left on the screen; displays the next 36 selected filenames until EOF (end of file).
R	Restart - redisplays the current screen as it was before any erasure(s).
W	Wrap - stores the remaining filenames for subsequent processing. It restarts the selection process by renumbering files that were not deleted and wrapping to the beginning of the workfile. Any files listed on a page not reviewed are not processed.
nn	nn - the number assigned to a selected file. Erases filenames from screen and files will not be copied.
?	Call out help module.

If the files to be erased have consecutive numbers on the screen, the range may be specified by placing a hyphen between the beginning and ending number (i.e., 3-15).

Several numbers may be entered at the same time but they must be separated by a space or comma.

To erase files 2, 5, 9, 12, 13, 14, and 30 from the screen, the user may enter

5, 9, 12-14 30 2

INDEX

INDEX

COMMENT SHEET

MANUAL TITLE IRIS R8 Operating System User Manual

PUBLICATION NO. SM-030-0011 REVISION 04

FROM: NAME/COMPANY: _____

BUSINESS ADDRESS: _____

CITY/STATE/ZIP: _____

COMMENTS: Your evaluation of this manual will be appreciated by POINT 4 Data Corporation. Notation of any errors, suggested additions or deletions, or general comments may be made below. Please include page number references where appropriate.

NO POSTAGE NECESSARY IF MAILED IN U.S.A.
Fold on Dotted Lines and Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1458 TUSTIN, CA

POSTAGE WILL BE PAID BY ADDRESSEE

POINT 4 Data Corporation
PUBLICATIONS DEPARTMENT
15442 Del Amo Avenue
Tustin, CA 92680

CUT ON THIS LINE

15442 Del Amo Avenue
Tustin, CA 92680-6465
(714) 838-2225