IRIS R9

User Reference Manual

IRIS R9
USER REFERENCE MANUAL

Revision 05

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) 259-0777

REVISION RECORD

PUBLICATION NUMBER: SM-030-0034

Revision	Description	Date
01	Initial release to SQA (adapted from R8 User Manual) (9.0)	09/22/86
02	Update package to SQA	11/10/86
03	Complete revision to SQA (incorporates commands from Operations and other manuals)	01/09/87
04	Preliminary release to customers; corresponds to IRIS 9.0	02/16/87
05	Update package incorporating changes for IRIS 9.1	06/26/87

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
Cover	-
Title	05
ii	03
iii, iv	05
v thru vii	03
viii thru xii	04
1-1	03
1-2 thru 1-4	04
1-5	03
1-6, 1-7	04
1-8 thru 1-10	03
1-11	04
1-12	03
1-13	04
2-1	04
2-2, 2-3	03
2-4	04
2-5	03
2-6	04
2-7	03
2-8	04
2-9 thru 2-14	03
2-15	04
2-16, 2-17	03
2-18	04
2-19 thru 2-34	03
2-35	04
2-36	03
2-37	04
2-38	03
2-39	04
2-40 thru 2-53	03
2-54	04
2-55 thru 2-58	03
2-59	04
2-60 thru 2-67	03
2-68	04
2-69 thru 2-77	03
2-78	04
2-79	03
2-80	04
2-81 thru 2-90	03
2-91	04
2-92, 2-93	03
2-94	04
2-95 thru 2-100	03
2-101	04
2-102 thru 2-129	03
2-130	05
2-131, 2-132	04
2-133	05
2-134, 2-135	03
2-136	04
2-137 thru 2-151	03
2-152	04
2-153 thru 2-161	03
2-162	05
2-163, 2-164	03
2-165	04
2-166 thru 2-201	03
3-1 thru 3-6	03
3-7	04
3-8 thru 3-23	03
Appendix Title	-
A-1, A-2	03
A-3	04
A-4, A-5	03
B-1	04
B-2 thru B-4	03
B-5	04
C-1 thru C-14	03
D-1	04
D-2	03
D-3	04
E-1 thru E-8	03
Index-1 thru Index-5	04
Comment Sheet	05
Mailer	-
Back Cover	-

The IRIS R9 User Reference 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 commands are presented in alphabetical order for easy reference.

The appendices contain an IRIS glossary, a description of terminal command keys and list of ASCII codes under IRIS, lists of IRIS components, error code lists and BUILDPF exercises.

For specific instructions on the installation and configuration of an IRIS system, please refer to the IRIS R9 System Configuration Manual. Daily operations such IPL, backups, and the general maintenance of the system are discussed in the IRIS R9 System Manager 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	Underlined information indicates user input. The input 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 R9 System Configuration Manual	SM-030-0029
IRIS R9 System Manager Manual	SM-030-0030
IRIS R9 Business BASIC Manual	SM-030-0028
IRIS R9 Peripherals Handbook	SM-030-0032
IRIS R9 Release Notes	SM-030-0033
MARK 5/9 Computer Reference Manual	HM-085-0032
POINT 4 SMbasic Manual	AM-190-0027
Thoroughbred BASIC Reference Manual	

CONTENTS

APPENDICES

INDEX

FIGURES

TABLES

Section 1

INTRODUCTION AND IRIS CONVENTIONS

1.1 INTRODUCTION

IRIS is an acronym for Interactive Real-Time Information System, POINT 4's operating system. IRIS is a high-performance system that supports an interactive, multiuser, timesharing environment.

IRIS supports two BASIC language interpreters, one for IRIS Business BASIC and one for IRIS SMbasic; IRIS also supports a preprocessor for IRIS Business BASIC called ABASIC. 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 data file types include text, formatted, indexed, contiguous, and polyfiles. Refer to the IRIS R9 Business BASIC Manual for a detailed description of IRIS data 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 R9 Business BASIC Manual contains a list and a description of these error codes. The Thoroughbred BASIC* Reference Manual and POINT 4's SMbasic Manual contain information about SMbasic error codes. System halts and traps are described in the IRIS R9 System Manager Manual.

*Thoroughbred BASIC is a trademark of Concept Omega Corporation.

1.2 USER ACCOUNTS

Each user is assigned an account by the IRIS system manager using the IRIS utility, ACCOUNTUTILITY. 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. 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 IRIS system manager. The Account ID string may consist of upper and lowercase characters. It must be entered exactly as defined by the 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 all level 0 or 1 files; has limited access to certain system files

  3 - may access all files (restricted to IRIS manager account)

- Account number - identifies files created by a user; consists of a group number and a user number within that group.

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

  Connect time is charged 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, at which time a message reporting the overtime condition is displayed. No user will be allowed to log on to that account until the IRIS system manager revises the time limitation.

- Assigned priority - a priority from 1-7 is assigned to each account by the IRIS 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 IRIS 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 (if any) 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 IRIS system manager for analysis.

- Assigned logical unit - a user is assigned to a logical unit (LU) by the IRIS 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 also be allotted disk blocks on other logical units.

For information on assigning accounts, see ACCOUNTUTILITY in Section 2.

1.3 USER TERMINALS UNDER IRIS

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

1. Idle state - the terminal is in an idle state when it is not logged on.

   The terminal can be logged on by pressing <ESC> any time that IRIS is active on the system. When <ESC> is pressed, a welcome message is displayed, followed by a prompt for 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 IRIS system prompt (#). In this mode, IRIS system commands such as BASIC, SAVE, KILL, LIBR, BYE, etc., can be entered.

   Most utilities have been given a verb as a filename that can be entered as a system command. In this manual, the utility filename is shown in uppercase letters, although it may be entered in lowercase letters unless otherwise noted. To enter a filename, system command, or a response to a prompt, <RETURN> must be pressed unless otherwise noted.

3. Processor mode - after a system command is entered, the terminal is in processor mode. (Most system or language commands invoke either a machine code program called a processor or an IRIS Business BASIC program.) Some utilities, 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 FILES

IRIS files are categorized by file type based on the function of the information in the file. Table 1-1 lists the file type codes and functions.

Data files are discussed in detail in the IRIS R9 Business BASIC Manual.

TABLE 1-1. FILE TYPE CODES

Octal Code	Letter Code	Description
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	Y	Polyfile
34		Reserved
35		Reserved
36	$	Peripheral device driver
37		Reserved

1.4.1 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: drivers 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 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:

```
R9.ACCTS.1
XY14.99
AB.123C
RESEARCH<CTRL-E>YY
$LPT
```

Each filename on a logical unit must be unique, but the same filename may be used on different logical units.

To access a program or utility, enter the following at the IRIS system prompt (#):

filename

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

To reference a file on a logical unit other than the user's assigned LU, the logical unit number must be included with the filename:

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.

To reference a specific volume of a polyfile, the volume number must be appended to the end of the filename and placed within brackets.

Examples of volume references are:

```
BIGDATAFILE[0]      (refers to volume 0)
BIGDATAFILE[8]      (refers to volume 8)
BIGDATAFILE[39]     (refers to volume 39)
```

1.4.2 File Cost and Protection

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

To specify cost when creating a file, precede the filename with the amount to be charged in the format $ddd.cc. The default for cost is zero.

The protection level code is specified as 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. Table 1-2 lists protection codes; the codes may be combined to provide more than one type of protection. For example, copy- and write-protection can be combined by specifying protection level 3. 77 gives maximum protection and is the system default.

To specify protection when creating a file, precede the filename with the protection code enclosed in angle brackets. For example, to create a file with protection 33 (copy- and write-protected against users at all levels), specify the filename as follows:

```
<33> filename
```

Cost and protection may be changed or added to any file by using the CHANGE command. For polyfiles, the cost and protection of the master volume determines the cost and protection for the entire polyfile.

Cost and protection are optional.

TABLE 1-2. PROTECTION LEVEL CODES

Code	Description
0	No protection
1	Copy-protect
2	Write-protect
4	Read-protect

1.4.3 Replacing Files

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

Polyfiles and polyfile volumes may not be replaced.

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>.

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

   - o Press <CTRL-D> (on any terminal)
   - o Press <CTRL-[> (some terminals)
   - o 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 IRIS 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 IRIS system manager has the option of suppressing some or all of this account information.)

4. Depending on the configuration of the system, the IRIS system prompt (#) or an application screen is displayed. This indicates that log-on has been completed.

```
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

1.5.1 Log-On Problems

A log-on problem is any event that does not allow the user to log on to the system. The problem may be that the system does not respond to the <ESC> key, does not accept a valid Account ID, or displays spurious characters such as @.

The following are log-on problems and possible solutions:

- o The system does not respond to the <ESC> key:
  - The timesharing system may be shut down for use in stand-alone mode by the IRIS system manager (for example, for maintenance or backups).

    If the system is in stand-alone mode, wait until operations return to normal.
  - The terminal may not be connected properly to the computer, or the connections may be defective.

    Check the cable connections; if necessary, request assistance from maintenance personnel.
  - System hardware may be causing a malfunction.

    Request assistance from maintenance personnel.
- o The system does not accept a valid Account ID:
  - The parity of the system may be incompatible with that of the terminal.

    To check for a parity problem, enter the letters ABD followed by the letters CEF. If the terminal beeps after each character in one group and does not beep for any character in the other group, the parity is not set correctly for the port.

    Change the parity setting by pressing <CTRL-P>. A per cent sign (%) is displayed. Press <ESC>, then reenter the Account ID.
- o The Account ID is displayed as it is being entered:
  - Press <CTRL-E>, then reenter the Account ID.
  - If this does not solve the problem, the terminal may be in half-duplex mode instead of full-duplex mode.

    Check the terminal documentation to determine the method for setting full-duplex mode.

- Spurious characters, such as @, are displayed when <ESC> is pressed:

  - This usually indicates that the baud rates for the terminal and system are not compatible.

    Have the IRIS system manager determine the baud rate that the system expects, then check the terminal documentation to determine how to set the terminal baud rate.

    If the system has a POINT 4 MIGHTY MUX and the auto-frequency scan bit is set in the port's Port Control Word (PCW), it is possible to select the next baud rate by pressing the <BREAK> key, then <ESC>. Continue to do this until the log-on prompt is displayed. If the log-on prompt is not displayed after going through all the available baud rates, check with the IRIS system manager. For more information on the auto-frequency scan bit and PCW, see the IRIS R9 System Configuration Manual.

    If the system is up, the baud rate can be changed by using the PORT BAUD n command (see PORT in Section 2).

  - If the baud rate is set correctly, the hardware may be malfunctioning.

    Request assistance from maintenance personnel.

- 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 IRIS system manager (i.e., uppercase, lowercase, or both).

  - If the problem continues, verify your Account ID with the IRIS system manager.

- The system prints the message

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

  Your assigned logical unit (LU) has not been installed. Notify the IRIS system manager that your LU is not installed. After installation, repeat the log-on process.

o One of the following error messages is 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)

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.

Check with the IRIS system manager.

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 IRIS system prompt (#), enter

BYE

Generally, the account status is then displayed as shown in Figure 1-2. The IRIS 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. (For more information on the Port Control Word, see the IRIS R9 System Configuration Manual.)

```
#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

This section covers the system commands and utilities available with an IRIS system. The section is arranged in alphabetical order for easy reference.

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

- Entry is not the name of a program or processor.
- Syntax is incorrect.
- Entry is not on logical unit 0, the default logical unit, nor the user's assigned logical unit; or, if the logical unit was specified, it is not on the specified logical unit.
- Utility or processor is protected against this user.

If the command is rejected, a message similar to the following is displayed, which describes the problem:

```
? NO SUCH PROCESSOR
```

The IRIS system prompt (#) is displayed.

Commands, debuggers, and utility programs used for system configuration procedures are described in the IRIS R9 System Configuration Manual and IRIS R9 System Manager Manual. Changing passwords and program options is discussed in the IRIS R9 System Manager Manual. Commands that are activated via a language processor, such as BASIC's NEW, SIZE or RENUMBER, are discussed in the appropriate language manual. For a complete list of IRIS components and the name of the manual in which each component is documented, see Appendix C, IRIS Components.

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 R9 Business BASIC Manual.

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 IRIS system prompt (#):

ABASIC {source filename}

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

For information on using the ABASIC preprocessor, refer to the IRIS R9 Business BASIC Manual.

ACCOUNTUTILITY

Each user on an IRIS system is given an account that is identified by a group and user number. The account is accessed by entering the Account ID assigned by the IRIS system manager when the account was set up. The information required to set up a user account is described in Table 2-1.

The utility program called ACCOUNTUTILITY is used to set up and maintain user accounts. The program is interactive and guides the user through the required functions by the display of various menu selections and appropriate prompts. ACCOUNTUTILITY can be used only from the IRIS manager account.

User account information exists in at least two places: logical unit 0 and the user's assigned logical unit. As the required information for the fields shown in Table 2-1 is entered, the program assigns the input to the ACCOUNTS files on logical unit 0 and the logical unit assigned to the user. However, the allotted disk blocks are assigned only to the user's assigned logical unit. After the account information has been entered, the program requests other logical units on which the user is to be assigned blocks. An account is set up on each logical unit on which space is assigned to the user.

TABLE 2-1. USER ACCOUNT INFORMATION

Description	Type	Range
Account ID	Alphanumeric	Up to 12 char
User Name	Alphanumeric	Up to 14 char
Privilege Level*	Numeric	0-2
Account Number		
Group	Numeric	0-255
User	Numeric	0-63
Assigned Priority	Numeric	1-7
Connect Time	Numeric/U**	0-1000 hrs or U
CPU Time	Numeric/U	0-1000 hrs or U
Assigned Logical Unit	Numeric	0-127 (must be active)
Disk Blocks (on Assigned LU)	Numeric/U	0-65535 or U

*The privilege levels include the following settings:

0 - lowest level; may access own files and all files not protected against such use

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

2 - privileged level; may access all level 0 and 1 files and all files not protected against such use; includes the UTILITY account, group and user 0,2

3 - the MANAGER account; may access all files, regardless of protection; this account is preset by POINT 4 as group and user 0,1; only one level 3 account may be present on a system

**The character U indicates a request for unlimited assignment.

ACCOUNTUTILITY Menu

To invoke ACCOUNTUTILITY, enter the following at the IRIS system prompt (#):

```
ACCOUNTUTILITY
```

The Accounts File Maintenance Menu is displayed:

```
ACCOUNTS FILE MAINTENANCE    REV n.n

(0) EXIT THE SYSTEM
(1) ADD NEW ACCOUNT
(2) MODIFY ACCOUNT
(3) DELETE ACCOUNT
(4) INQUIRE ACCOUNT
(5) LIST THE ACCOUNTS

ENTER FUNCTION NUMBER: _
```

Each option is discussed in the following pages.

Throughout ACCOUNTUTILITY, <ESC> may be used as follows:

- At the first input field of the screen, <ESC> may be used to return to the previous menu.
- At subsequent input fields, <ESC> may be used to back up to the previous input field.
- At the Accounts File Maintenance Menu, <ESC> may be used to return to the IRIS system prompt.

Specifying an Account

A logical unit number and account must be specified when the modify, delete, and inquire functions are selected. Prompts similar to the following are displayed:

```
ENTER LOGICAL UNIT:
```

Enter the logical unit number. The logical unit must be active.

The program then displays a menu which contains the methods of specifying an account, similar to the following:

```
ACCOUNT function ON LU#n

(0)   RETURN TO MAIN MENU
(1)   SELECT BY RECORD NUMBER
(2)   SELECT BY ACCOUNT GROUP, USER
(3)   SELECT BY ACCOUNT ID
(4)   SELECT BY USER NAME

ENTER FUNCTION NUMBER:
```

Function may be MODIFICATION, DELETION or INQUIRY.

NOTE

Options 3 and 4 are available only when retrieving information from logical unit 0.

To return to the Accounts File Maintenance Menu, select option 0.

To retrieve an account by record number, select option 1. If the record is not in the file (e.g., the record number given was wrong or it had been deleted) the program responds

```
RECORD nnn NOT FOUND, TRY AGAIN !
```

To retrieve an account by group and user numbers, select option 2. If the selected option is modify and the account is not found on the specified logical unit, but the account exists on logical unit 0, the program responds

```
g,u NOT FOUND
ACCOUNT FOUND ON LU#0
ADD THE ACCOUNT TO LU#n ? (Y/N):
```

Enter Y to add the account. Enter N to select another account.

If the account does not exist on any logical unit, the program responds

```
g,u NOT FOUND, TRY AGAIN !
```

To retrieve an account by account ID, select option 3. If the account is not found, the program responds

```
account id NOT FOUND, TRY AGAIN !
```

To retrieve an account by user name, select option 4. If the user name is not found, the program responds

```
user name NOT FOUND, TRY AGAIN !
```

If logical unit 0 is specified, the current status of the account is displayed as shown in Figure 2-1; if a nonzero logical unit is specified, the status is displayed as shown in Figure 2-2.

```
                ACCOUNT STATUS ON LU#0

                ACCOUNT CREATION DATE:  11/12/86
                RECORD NUMBER:          3
(I)             ACCOUNT ID:             USER1
(N)             USER NAME:              name
(L)             PRIVILEGE LEVEL:        1
(A)             ACCOUNT GROUP, USER:    1, 1
(P)             ASSIGNED PRIORITY:      5
(M)             CONNECT TIME REMAINING: UNLIMITED
(S)             CPU TIME REMAINING:     UNLIMITED
(U)             ASSIGNED UNIT:          3
(D)             DISK BLOCKS ALLOTTED:   0
                DISK BLOCKS IN USE:     0
(C)             TOTAL FILE USE CHARGE:  $000.00
```

Figure 2-1. Account Status on Logical Unit 0

```
                ACCOUNT STATUS ON LU#u

                RECORD NUMBER:          4
(A)             ACCOUNT GROUP, USER:    1, 1
(D)             DISK BLOCKS ALLOTTED:   10000
                DISK BLOCKS IN USE:     23
```

Figure 2-2. Account Status on a User Logical Unit

Adding Accounts

Option 1 from the Accounts File Maintenance Menu invokes the new accounts module. The program prompts for the information shown in Table 2-1. <ESC> may be used to back up to the previous entry field.

After the last field has been entered, the account status is displayed. The following prompt is then displayed:

```
UPDATE THE ACCOUNT FIELDS ? (Y/N):
```

Enter Y to add the account. Enter N to reenter the fields and make any necessary corrections.

After Y has been entered, the program responds

```
UPDATING ACCOUNT FIELDS ON LU#n
UPDATING ACCOUNT FIELDS ON LU#0
```

where
n - number of the assigned logical unit

The program then prompts for disk block allotments on other logical units as follows:

```
ALLOT DISK BLOCKS ON OTHER LU
ENTER LU/DISK BLOCKS (U=UNLIMITED):
```

To set up disk blocks on additional logical units, enter the logical unit number, including the slash (/), followed by the number of disk blocks to allocate. (The logical unit must be active.) To specify unlimited usage, enter U as the number of disk blocks. For example, to allocate unlimited usage on logical unit 3, enter 3/U.

After disk block usage has been entered for all additional logical units, press <ESC>. The program updates the account while displaying

```
UPDATING ACCOUNT FIELDS ON LU#n
```

The Accounts File Maintenance Menu is displayed.

If the account already exists on a specified logical unit, a message similar to the following is displayed:

```
ACCOUNT EXISTS ON LU#n, NOT UPDATED
```

The program then returns to the LU/Disk Blocks prompt.

Modifying an Account

Option 2 from the Accounts File Maintenance Menu invokes the change module, which allows modification of an existing account.

After the account is specified and displayed (see Specifying an Account), the following prompt is displayed:

```
ENTER FIELD LETTER, <RETURN> WHEN DONE:
```

Enter the letter of the field to modify. The program displays the field content. Enter new information or press <RETURN> for no change. The Field Letter prompt is repeated. When all modifications have been entered, press <RETURN> at the Field Letter prompt without entering any data.

The following prompt is displayed:

```
UPDATE THE ACCOUNT FIELDS ? (Y/N):
```

Enter Y to accept the modifications. Enter N to retain the account as is.

The program then asks for another account to be modified. To return to the Account Modification Menu, press <ESC>.

Deleting an Account

Option 3 from the Accounts File Maintenance Menu invokes the module that allows deletion of an account.

After the account is specified and displayed (see Specifying an Account), the following prompt is displayed:

```
ACCOUNT g,u EXISTS ON THE FOLLOWING ACTIVE UNITS:
a; b; c; ...

DELETE THE ACCOUNT FROM ALL ACTIVE UNITS ? (Y/N): _
```

To delete the account from all listed units, enter Y. To retain the account on the listed units, enter N.

If N is entered, the program asks

```
DELETE THE ACCOUNT FROM LU#n ? (Y/N):
```

To delete the account from the specified unit, enter Y. To retain the account, enter N.

The program then requests another account number. To return to the Accounts File Maintenance Menu, press <ESC>.

Querying an Account

Option 4 from the Accounts File Maintenance Menu invokes the module that allows examination of a user account on a specified logical unit.

After the account is specified and displayed (see Specifying an Account), the following prompt is displayed:

```
PRESS <RETURN> TO CONTINUE
```

The program then requests another account. To return to the Account Inquiry Menu, press <ESC>.

Listing Accounts

Option 5 from the Accounts File Maintenance Menu allows the listing of all the accounts on a specific logical unit. It gives the option to print or display the listing. The first prompt is for logical unit:

```
ENTER LOGICAL UNIT:
```

Specify the appropriate LU; the program then prompts for the type of listing or display

```
SELECT OUTPUT 1=DEVICE 2=FILE <RETURN>=CRT:
```

where

1 - outputs the report to a specified device. The following prompt is displayed:

```
ENTER DEVICE NAME, <RETURN>=$LPT:
```

The input must begin with a dollar sign ($). If an error occurs it will be reported and the program will repeat the device name prompt. Press <ESC> to return to the output selection prompt.

2 - outputs the report to a specified text file. The following prompt is displayed:

```
ENTER LU/FILENAME:
```

The program will try to build the file. If it already exists, the user must include an exclamation point (!) at the end of the filename to overwrite the existing file. Press <ESC> to return to the output selection prompt.

<RETURN> - outputs the report to the terminal.

The display is similar to the following:

```
MAR 17, 1987  10:57:49                                        PAGE  - 1

                          ACCOUNTS LIST ON LU#0

CREATED    RECD  ACCOUNT   USER        ACCOUNT      USER   ASGN   ALLOTTED
 DATE      NO.     ID      NAME     PRV  GRP USR    PRI    LU#    BLOCKS

10/10/86    1    MANAGER  NO NAME   3    0    1     7      0     UNLIMITED
10/10/86    2    UTILITY  NO NAME   2    0    2     5      0     UNLIMITED
11/12/86    3    USER1    name      1    1    1     5      3           0
11/12/86    4    USER2    name      1    2    1     5      4           0
12/01/86    5    USER3    name      0    5   24     4      8           0
```

For logical units other than 0, the entries for account ID, user priority (USER PRI), and assigned logical unit (ASGN LU#) are left blank.

After the accounts have been listed, the following is displayed:

```
PRESS <RETURN> TO CONTINUE
```

To select another logical unit, press <RETURN>.

To return to the Accounts File Maintenance Menu, press <ESC>.

ACS.TDCOPY

ACS.TDCOPY is a POINT 4 diagnostic tool and is not intended for customer use.

ACS.VERIFY

ACS.VERIFY is a POINT 4 diagnostic tool and is not intended for customer use.

ALOAD

ALOAD is a utility program that may be used to merge an assembly language file into an existing file such as REX or DISCSUBS. ALOAD does not overwrite cells for which the IRIS Assembler has not generated any values in the source file (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 processor or program (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 and should be saved on a manager or utility account at protection level 77.

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

```
ALOAD
```

The program then displays

```
ALOAD - * A FILE LOAD

File to load from:
```

Enter the number of the source logical unit 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 number of the destination logical unit followed by the desired filename. If the file is found, ALOAD displays

```
Loading .....
  x-y
```

where

x - beginning address currently being written
y - ending address currently being written

Upon successful completion of the procedure, the IRIS system prompt (#) is displayed.

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

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

ANALYPF

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

To find the correct volume number of the directory to be analyzed, use the QUERYPF functions accessible from ANALYPF. Refer to the section, QUERYPF, for information on using those functions.

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

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 not related to a directory volume number.

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.

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).

Errors may be generated by two functions used by ANALYPF: CALL $VOLLINK and SEARCH. For a list of error codes, see Appendix D.

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 IRIS system 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 [0-39]
     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, directories that 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 [0-39]
     or   <RETURN> for global display
     or      -1 for complete dump
     or      -2 for analysis mode

      or ESCape to exit to SCOPE: -2

Volume # (octal): 0
Block # (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 ***

Block # (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 ***
    Block # (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:

```
Block # (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 R9 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:

```
Block # (octal): <RETURN>
Volume # (octal): <RETURN>
Scanning polyfile, please wait...
Please input volume number [0-39]
     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:

```
Block # (octal): 1

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

Block # (octal): 0

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

Block # (octal): <RETURN>
Volume # (octal): 0

Block # (octal): 0

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

Block # (octal): <RETURN>
Volume # (octal): <RETURN>
Scanning polyfile, please wait...
Please input volume number [0-39]
     or   <RETURN> for global display
     or      -1 for complete dump
     or      -2 for analysis mode

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

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 MARK 5/9 Computer Reference Manual. Information on permissible opcode parameters is contained in the 0/SYMBOLS text file. This file can be printed by copying it to a printer.

Any IRIS editor can be used to create an ASM program in text file format.

To list or assemble an ASM program, at the IRIS system 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

The specified source files are processed in sequence.

*Nova is a trademark of Data General Corporation.

BASIC

IRIS Business BASIC is one of two BASIC programming languages supported under IRIS. For a detailed description of IRIS BASIC, refer to the IRIS R9 Business BASIC Manual.

BASICTEST

BASICTEST is a software diagnostic program that checks whether the statements in IRIS Business BASIC are functioning correctly. BASICTEST can be used only from the IRIS manager account, or accounts in group 0, user 2 and 3.

Before running BASICTEST, logical unit 1 must be installed and a text file copy of the BASICTEST program must be placed on logical unit 1 and given the name BT. (the period is part of the name). The BASIC statement DUMP can be used.

To dump, then run BASICTEST, enter the following lines:

```
#BASIC BASICTEST
DUMP 1/BT.
RUN
```

To run BASICTEST from the IRIS system prompt (#), enter the following:

```
#BASICTEST
```

If the test runs successfully, the time to run the test is displayed, similar to the following:

```
 TIME = 11.6
```

If an error is detected, the program stops and displays the line number that contains the stop. List the program to determine the function in which the error occurred.

BCONVERT

BCONVERT is used to convert IRIS Business BASIC programs from IRIS R7 format to IRIS R9 format.

For more information, see the IRIS R9 Release Notes.

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. For a more detailed explanation of polyfiles, see the IRIS Business BASIC Manual. A BUILDPF exercise is provided in Appendix E.

Summary of Polyfile Features

A polyfile is made up of one or more associated volumes tied together by the filename, last access date, and pointers contained in a master volume header. The maximum number of volumes is 40 (numbered 0-39); the master volume is always volume 0.

Volumes of a polyfile may reside on different logical units, except that no polyfile may reside on logical unit 0.

The various types of volumes are the following:

- Base directory - contains the master block and first fine block for one or more directories
- Directory extension - provides additional disk blocks that are used to extend a base directory volume
- Data volume - contains data records; the same record size is used for all data volumes in a given polyfile; records are numbered sequentially through ascending data volume numbers

  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

Points to remember when building a polyfile include the following:

- Use the map option for data volumes to allow the system to keep track of free records. If the map option is not used, the free records must be tracked by the user program. In any one polyfile, all data volumes must be either mapped or unmapped.
- The size of a volume is limited by the size of the logical unit 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 = 65535*(r)/256 (to a maximum of 65535 blocks)

  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 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.

- Keys in the directory volumes (base or extended) may be from 2 to 121 bytes in length.

- 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.

- Errors may be generated by two functions used by BUILDPF: CALL $VOLLINK and SEARCH. For a list of error codes, see Appendix D.

Using BUILDPF

To use BUILDPF, at the IRIS 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)]:
```

The range of possible LUs is from 1 to 127. The filename may contain up to 13 characters.

BUILDPF then attempts to open the file. If the polyfile is found, BUILDPF enters the polyfile extension mode (described later in this section). 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.

NOTE

If an invalid response is entered, the prompt is redisplayed. No error message is given.

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:
```

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 - specified starting directory number
yy - range of contiguously available directory numbers

BUILDPF then prompts for the key size for the volume as follows:

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

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.

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.

Volume Size Input

The volume size information required 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.

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 (see Base Directory) 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 number of blocks to be allocated by the specified key size.

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 WILL HOLD A MAXIMUM OF APPROXIMATELY kkkk KEYS.

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

If the size is given in keys, enter the number of additional keys for the base directory plus its directory extensions. 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 size is given in blocks, the size applies only to the volume being defined.

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.

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 - 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 Volume Type Input). If N is entered, the user is given the option to build another type of 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.

Polyfile Extension Mode

If the specified file has been created previously, 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 [0-39; <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 (see Volume Type Input) and volume size (see Volume Size Input), before building and structuring the volume.

Press <RETURN> to exit the program; the IRIS system prompt (#) is then displayed.

BUILDPFERR

BUILDPFERR is used to create a file on logical unit 0 called POLYFILERRORS that contains error messages used by ANALYPF, BUILDPF, and QUERYPF. The user must have blocks allocated on logical unit 0 in order to create the file.

To invoke the program, enter the following at the IRIS system prompt (#):

```
BUILDPFERR
```

The file is built and the following message is displayed:

```
Polyfile error file built.
```

For a list of the error messages, see Appendix D, Error Codes.

BUILDXF

BUILDXF is an interactive BASIC program used to build indexed files. 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:
```

- o Number of data records - number of actual data records to be contained in the file. The records are put on the free list. For a directory-only file, enter 0.

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

- o Number of indexed records - number of keys in directory with the greatest number of keys. If the number of keys varies significantly from one directory to another, the directories can be built as separate indexed files to avoid wasting disk space. This requires one channel per indexed file.

- o Number of directories - total number of directories required.

- o 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 the directories specified.

The program displays the actual number of available data records which could 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 program displays an appropriate message. Contiguous space may be recovered by one of the following:

- o 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 the utility CLEANUP to reorganize the logical unit
- Using INSTALL AND CLEAR to create a new logical unit; this deletes all files and accounting information on the LU

Summary of Indexed File Features

An indexed file contains a number of blocks that serve as directories. A maximum of 15 directories is allowed.

The directories may point to the data portion of the file; they may point to records contained in other files, or the directory may comprise the entire file.

Each directory of an indexed file has three levels:

- o Master
- o Coarse
- o Fine

The master and coarse levels contain pointers to the next level, and the fine level contains pointers to the associated data 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 record number of the first real data record 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

Because 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 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 maximum possible number of data records in an indexed file is determined by the number of blocks available on the logical unit, less those used for the directories.

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 R9 Business BASIC Manual for more information on indexed files under IRIS.

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

TABLE 2-2. 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	61225
5	42	36141
6	36	22662
7	31	14400
8	28	10570
9	25	7488
10	23	5808
11	21	4400
12	19	3240
13	18	2745
14	16	1912
15	15	1568

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

Using BUILDXF

To use the BUILDXF command, at the IRIS 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 BUILDXF has finished structuring the file, the IRIS system prompt (#) is displayed.

The QUERY command (see QUERY) 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

PRIV: 2,  ACCOUNT GROUP: 1,   USER: 2, UNIT: 3
PROTECTION: 77,  SIZE: 47 BLOCKS,  AGE: 0 HOURS
LAST ACCESSED:  0 HOURS AGO,   COST:   $0.00,  TOTAL INCOME:  $0.00
```

Creating a Directory-Only File

The procedure for creating a directory-only file is the same as shown above, except the response to the Number of Data Records? prompt should be zero.

BYE (LOG OFF)

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

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

At the IRIS system prompt (#), enter

```
BYE
```

Some or all of the following information may be displayed:

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

NET ACCRUED CHARGES      $dd.cc

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

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

For information on changing the display of information, see the IRIS R9 System Manager Manual.

CHANGE

CHANGE is a system command used to change certain file characteristics. The user who created a given file, has write-access to the file, or is signed on to the IRIS manager account may change the following:

- o Filename
- o Cost
- o Protection
- o A stand-alone program's starting address

For information on filenames, see Section 1.4.1; for information on cost and protection, see Section 1.4.2.

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

- o R, L and I control bits
- o Processor type
- o Priority

Changing File Characteristics

CHANGE prompts for each characteristic to be changed in the following sequence:

- o Name
- o Cost
- o Protection
- o RLI bits (if user privilege is 2 or greater and file type is less than 20)
- o Processor type (if user privilege is 2 or greater and R bit is set)
- o Starting address (if R bit is not set and file type is 3 or 4)
- o Priority (if user privilege is 2 or greater)

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.

CHANGING A FILENAME

To invoke CHANGE, at the IRIS system 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>.

CHANGING 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.

CHANGING 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 CHANGE is accessed from the IRIS manager account, additional options are displayed; see Manager Options for CHANGE.

CHANGING A FILE'S STARTING ADDRESS

If the file is not a runnable processor (the R bit is not set), and the file is a type 3 or 4, CHANGE displays

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

where

nnnnn - current starting address in octal

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).

Manager Options for CHANGE

The CHANGE options available to users signed on to the IRIS manager account allow a processor to be assembled and then changed, thus making it accessible for system use.

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

- o Change control bits
- o Change processor type
- o Change a file's priority

When a filename is entered in response to the IRIS system prompt (#), the file's header is read. The control bit in the header's TYPE word is then examined. Figure 2-3 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 initialized 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.

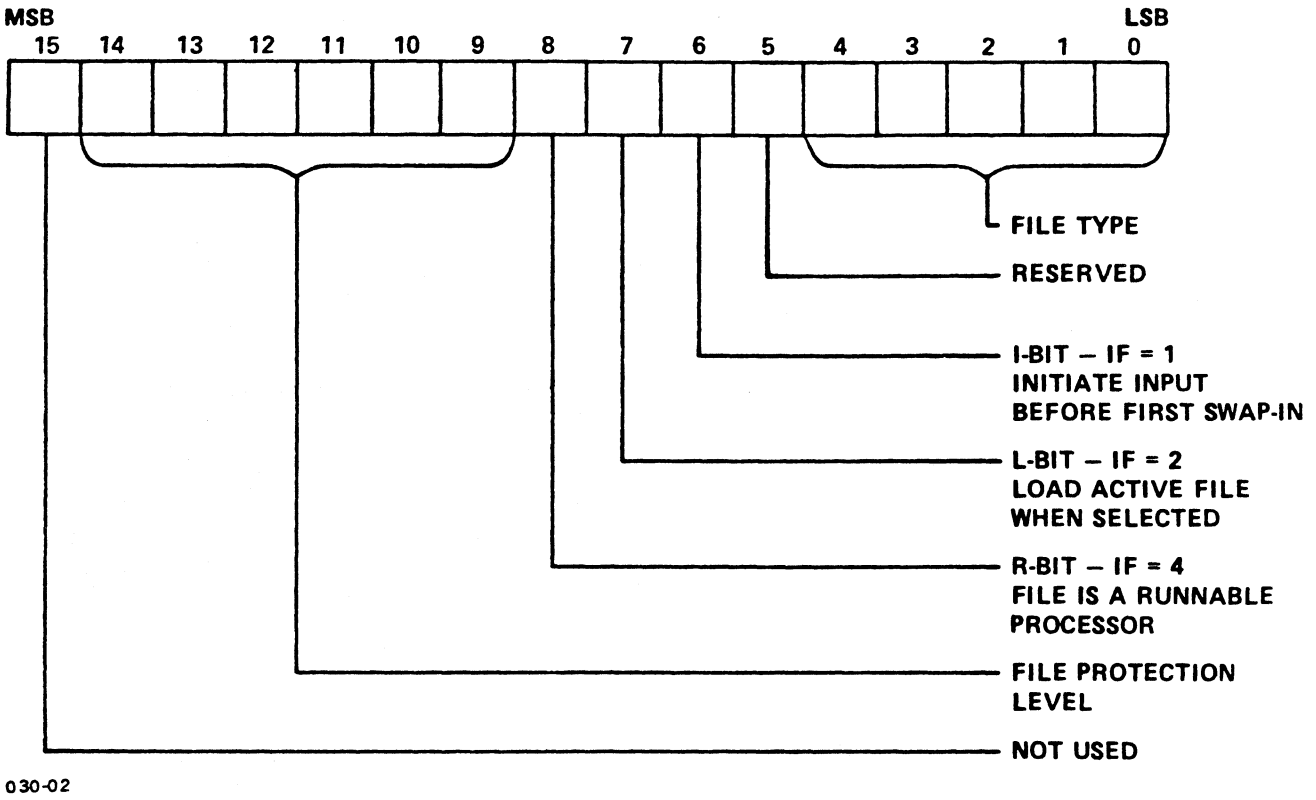

Figure 2-3. Control Bits in the File Header TYPE Word

CHANGING CONTROL BITS

This option permits a change of control bits; 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:

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

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 2-3) 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. See Table 1-1 for file type codes.

CHANGING A FILE'S PRIORITY

This option permits changing the priority of a file. CHANGE displays

```
PRIORITY = n
NEW PRIORITY?
```

where

n - current priority; if priority has never been defined, UNDEFINED is displayed

Priority values range from a low of 1 to a high of 7; the default is 5. If the file's priority is not in the range 1-7, the word UNDEFINED is displayed. At present, priority is meaningful only for IRIS Business BASIC programs and peripheral drivers.

Program priority is used by the system scheduler to determine the number of time slices to give to a program, relative to other programs. Programs with a priority or 1 or 2 are assigned to background mode and are given time slices only when there is no higher priority program waiting for a time slice. (For more information on the effect of priorities and time slices, see the IRIS R9 System Configuration Manual.)

Driver priorities are used by the system initialization routine (SIR) during an IPL to determine the order in which to initialize the drivers. This priority can be used to ensure that multiplexer ports are initialized before phantom ports. For example, if the multiplexer driver has a higher priority than the phantom port driver, it is initialized first and assigned port numbers before phantom ports.

If the multiplexer driver and the phantom port driver have the same priority, the system assigns port numbers based on the position of the drivers in the logical unit index and by the header block addresses of the drivers.

CLEANUP

CLEANUP is a utility program that compresses files in order to allow more effective access and to make contiguous space available on a logical unit. It does this by relocating files and processors. CLEANUP can be used only from the IRIS manager account.

CLEANUP performs the following functions in 18 phases (see CLEANUP Phases):

- Analyzes the input command to verify that the specified logical units are available
- Scans all ports to make sure all users are logged off
- Calculates the size of the LU to be cleaned up
- Builds a scratch file called GARBACO on the scratch LU
- Zeroes out the DMAP on the LU to be cleaned up
- Scans the INDEX on file type priority
- Relocates file headers
- Stores the old and new RDAs in GARBACO
- Repositions used disk blocks in a new and more sequential order
- Removes the LU if it is a nonzero LU

CLEANUP should be run regularly, for example, once a week on a system that is heavily used. The time required to do this depends upon the type of disk, size, number of files, and the amount of reordering necessary. Juggling disk blocks takes the most time. Because the initial organization takes longer, the first run of CLEANUP requires more time. Typically, a 5K block moving arm disk takes 10 to 20 minutes when 50 percent full and one to two hours if 80 percent full.

CAUTION

If the system includes a POINT 4 LCM, CLEANUP may not be run while the LCM is active.

Preparations for Running CLEANUP

The following preparatory steps must be completed before CLEANUP can be initiated:

1. Ensure that all users are logged off the system.

2. Back up the system. If a trap occurs while CLEANUP is being run, some blocks may have been moved to new locations while others were not, making the logical unit useless; if the logical unit is LU 0 (the system disk), an IPL may not work. It is necessary to have a copy of the system that can be used to restore the original unit.

3. Report any known bad blocks on the logical unit to be cleaned up. Bad blocks may be entered during an Initial Program Load (IPL) for LU 0 or INSTALL sequence for all other LUs.

4. Determine the size required for a scratch LU (one that is blank or can be written over) and INSTALL it (see INSTALL, Structuring a New Logical Unit).

   The scratch LU is used by CLEANUP to reorder files and processors. A map of old and new real disk addresses (RDAs) is built and file headers are relocated. Files are then 'juggled' into a new and more sequential order.

   Calculate the size of the scratch LU to be used by adding one header block to the total number of blocks on the logical unit to be cleaned up divided by 256. For example

   - If the size of the LU to be cleaned up contains 1000 blocks (decimal), the calculation is

     1 + (1000/256) = 5

     If the total number of blocks on the LU is not equally divisible by 256, round the result upward.

   - If the size of the LU to be cleaned up contains 32000 blocks, the calculation is

     1 + (32000/256) = 126

   If LU 0 is being cleaned up, the scratch LU should contain at least 100 blocks. If the scratch LU is too small, CLEANUP may not reallocate the active files or discsubs correctly, and it will produce a TRAP.

Using CLEANUP

After the preparations for CLEANUP are completed (that is, ensuring that all users are logged off, reporting bad blocks, backing up the system, determining the size required for a scratch LU and INSTALLing it), run CLEANUP as follows:

At the IRIS system prompt (#), enter

```
CLEANUP <CTRL-E> key <CTRL-E> a USING b
```

where
- key - password assigned to CLEANUP (the default is X)
- a - number of LU to be processed
- b - number of scratch LU

When CLEANUP is completed, one of the following messages is displayed as appropriate:

- For nonzero LUs:

  ```
  CLEANUP DONE.  MUST REINSTALL LOGICAL UNIT.
  ```

  The logical unit must be reINSTALLed (see INSTALL, Structuring a New Logical Unit) because CLEANUP removes nonzero LUs once processed.

- For logical unit 0:

  ```
  END OF CLEANUP - WHEN SYSTEM HALTS, RE-IPL
  ```

  If LU 0 was cleaned up, perform an IPL to restore the system.

CLEANUP Error Messages

When CLEANUP is run, certain error conditions may occur. The error messages are described below:

ILLEGAL INPUT - Input command was entered incorrectly.

SAME LU - Logical unit to be processed and the scratch logical unit were not entered as separate logical unit numbers.

NOT ALL USERS LOGGED OFF - A port was found that was in use or not logged off.

"garbaco" FILE ALREADY EXISTS - A file with the same name as the scratch file to be created by CLEANUP already exists on the scratch logical unit.

OUT OF DISK SPACE - Not enough free disk blocks on the scratch logical unit to create a scratch file.

NO "FIXDIRECTORIES" DISCSUBS ON THE SYSTEM - Subroutine used to fix the directories of indexed files is not in the DISCSUBS file.

BAD FILE DHDR IN INDEX - A file named in the INDEX did not have a correct header address. This is a FATAL ERROR!!!

GAP IN "ACCOUNTS" FILE BLOCKS - The list of blocks in the header of the ACCOUNTS file has a gap in it. Do an IPL or INSTALL to use that logical unit.

CLEANUP Phases

CLEANUP functions are performed in 18 phases. Phases 1 through 3 are internal and are not displayed. The filenames processed in phases 4 through 18 are as follows:

Phase Number	Filename
4	BZUD
4	INDEX (header)
4	REX (page zero)
4	ACCOUNTS (header)
4	DMAP
4	Nesting Blocks
5	ACCOUNTS
5	INDEX
6	DISCSUBS, MESSAGES
7	SCOPE
7	CONFIG (header, first block)
7	SAVE
7	RUNMAT
7	BASIC
7	RUN
8	Active files
9	LIBR, ASM, EDIT
10	Contiguous and Indexed files
11	Formatted and Text files
12	Other processors
13	Enabled drivers ($-files)
14	BASIC programs, miscellaneous files
15	CONFIG (i.e. block two to the end)
15	REX
16	Updated file addresses in INDEX
17	Shuffle blocks as determined in phases 5-15
18	Correct Indexed File directories

CLEANUPX

CLEANUPX is an extended version of CLEANUP. It compresses files and performs the same functions as CLEANUP, but it also creates a special area for the DMAP so that DMAP can be contained on any disk. CLEANUPX does not increase the size of the logical unit or the INDEX. CLEANUPX can be used only from the IRIS manager account.

CLEANUPX should only be used when an IRIS procedure calls for it. For example, CLEANUPX must be run before transferring a nonzero logical unit from one system to another.

To invoke CLEANUPX, enter the following at the IRIS system prompt (#):

```
CLEANUPX <CTRL-E> key <CTRL-E> x USING y
```

where

key - password assigned to CLEANUP (the default is X)
x - number of LU to be processed
y - number of scratch LU

When CLEANUPX is completed, one of the following messages is displayed as appropriate:

- For nonzero LUs:

```
CLEANUPX IS DONE.  IF THIS LOGICAL UNIT IS TO BE
TRANSPORTED, COPY TO TRANSPORT MEDIA BEFORE INSTALLING
LOGICAL UNIT.
```

  The logical unit must be reINSTALLed (see INSTALL, Structuring a New Logical Unit) because CLEANUPX removes a nonzero LU once processed.

- For logical unit 0:

```
END OF CLEANUPX - WHEN SYSTEM HALTS, RE-IPL
```

  If LU 0 was cleaned up, perform an IPL to restore the system.

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:

- o Copy a file
- o Copy a file from one logical unit (LU) to another LU
- o Merge several files into a single file
- o Compare and verify data files
- o Page or depage a text file
- o Copy text file(s) to a printer
- o Copy an entire polyfile

The COPY command format is

```
COPY destfile=sourcefile
```

where

destfile - name of destination file
sourcefile - name of file to be copied

If a protection level is not specified, the system defaults to protection level 77.

For information on filenames, refer to Section 1.4.

To copy a text file to a printer, use the printer name as the name of the destination file. For example, the following command causes the text file on LU 3 to be printed on $LPT1:

```
COPY $LPT1 = 3/textfile
```

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 recovered by 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
- Running the utility CLEANUP to reorganize the logical unit
- Using INSTALL AND CLEAR to create a new logical unit; this deletes all files and accounting information on the LU

At the IRIS system prompt (#), enter

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

where
r - number of records
w - record length in words
! - replace 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 is 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.

Copying Polyfiles

A polyfile may be copied to a single destination logical unit, or all the volumes of the polyfile may be selectively placed on specific logical units. The destination filename cannot already exist; the use of an exclamation mark (!) to replace a polyfile is not supported.

To copy all volumes of a polyfile to a single destination logical unit, use the same syntax as for other files.

To copy all volumes to specified logical units, use the following syntax:

```
COPY dest[lul/a{,b{,...}}{;lu2/c{,d{,...}}{;...}}] = source
```

where

lul, lu2 - specified logical units

a,b,c,d - volumes of the polyfile; may be either a single volume or a range of volumes

The destination logical units must be enclosed in brackets following the filename. Volumes and ranges of volumes for the same logical unit must be separated by commas; entries for different logical units must be separated by semicolons. All volumes must be copied.

For example, to copy polyfile PFA (which consists of volumes 0-8) to polyfile PFB, placing some volumes on LU 5 and some on LU 6, the following could be entered:

```
COPY PFB [5/0-4,8;6/5-7] = PFA
```

This copies volumes 0-4 and 8 to LU 5; volumes 5, 6, and 7 are copied to LU 6.

Concatenating Several Files

Multiple sources may be combined into a larger file to concatenate the data. The following groupings are allowed:

- o ASCII sources such as text files
- o Binary sources (machine code files such as processors); 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
file1... - source files to be combined

Names of source files must be separated by commas.

The source files (file1, file2, etc.) are not affected by use of the COPY command.

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

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

If an existing file is to be replaced, the format is

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

Compare and Verify Data

The COPY command can be used to compare and 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 message is displayed.

NOTE

Compare and verify cannot be used on indexed files or polyfiles.

Page or Depage a Text File

The COPY command can be used to page or depage a text file by following the destination filename with a # character.

To page a text file, at the IRIS system 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

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 formfeed codes in the source text.

COREMAP

COREMAP is a BASIC program that displays or prints out a map of the current memory allocation. It is used after an IPL and only from the IRIS manager account.

To run COREMAP, enter the following at the IRIS system prompt (#):

```
COREMAP
```

The following is displayed:

```
#0/LIBR 0/@[COREMAPLIBRFIL]
#BASIC
20 RUN


CONSTRUCTING COREMAP SCRATCH FILE, PLEASE STAND BY
SORTING COREMAP SCRATCH FILE, PLEASE STAND BY

PRINT WHERE (NAME OR L OR L1 OR CR) ?
```

where

NAME - any valid IRIS text filename; permits storage of scratch file for later reference or printout

L - $LPT

L1 - $LPT1

CR - <RETURN> displays the contents of the scratch file at the terminal

Enter print selection.

To halt the display to the terminal temporarily, press <CTRL-S>. To continue the display, press <CTRL-Q>. To abort the program, press <ESC>.

Following transfer, display, or printout of the source file, the IRIS system prompt (#) is displayed.

Figure 2-4 illustrates a partial printout of a COREMAP scratch file.

```
                                IRIS CORE MAP


    0   (   200) <-- REX PAGE ZERO
  200   (   400) <-- PROCESSOR PAGE ZERO
  600   (   110) <-- SYSTEM INFORMATION TABLE
  710   (    60) <-- INTERRUPT VECTOR TABLE
  770   (    20) <-- CHARACTER QUEUE USED BY PCHAR IN REX
 1010   (  5716) <-- BEGINNING OF REX CODE
 6726   (  2444) <-- CALL TRANSLATE TABLE FOR SYSTEM SUBROUTINES (PART OF REX)
11372   (    33) <-- BPSP (BEGINNING OF PATCH SPACE, IE. END OF REX)
11425   (   104) <-- I/O BUFFER FOR PORT # 35
11531   (   104) <-- I/O BUFFER FOR PORT # 34
11635   (   104) <-- I/O BUFFER FOR PORT # 33
11741   (   104) <-- I/O BUFFER FOR PORT # 32
12045   (   104) <-- I/O BUFFER FOR PORT # 31
12151   (   104) <-- I/O BUFFER FOR PORT # 30
12255   (   104) <-- I/O BUFFER FOR PORT # 29
12361   (   200) <-- I/O BUFFER FOR PORT # 28
12561   (   104) <-- I/O BUFFER FOR PORT # 27
12665   (   104) <-- I/O BUFFER FOR PORT # 26
12771   (   104) <-- I/O BUFFER FOR PORT # 25
13075   (   104) <-- I/O BUFFER FOR PORT # 24
13201   (   104) <-- I/O BUFFER FOR PORT # 23
13305   (   104) <-- I/O BUFFER FOR PORT # 22
13411   (   104) <-- I/O BUFFER FOR PORT # 21
13515   (   104) <-- I/O BUFFER FOR PORT # 20
13621   (   104) <-- I/O BUFFER FOR PORT # 19
13725   (   104) <-- I/O BUFFER FOR PORT # 18
14031   (   104) <-- I/O BUFFER FOR PORT # 17
14135   (   104) <-- I/O BUFFER FOR PORT # 16
14241   (   104) <-- I/O BUFFER FOR PORT # 15
14345   (   104) <-- I/O BUFFER FOR PORT # 14
14451   (   104) <-- I/O BUFFER FOR PORT # 13
14555   (   104) <-- I/O BUFFER FOR PORT # 12
14661   (   104) <-- I/O BUFFER FOR PORT # 11
14765   (   104) <-- I/O BUFFER FOR PORT # 10
15071   (   104) <-- I/O BUFFER FOR PORT # 9
15175   (   104) <-- I/O BUFFER FOR PORT # 8
15301   (   104) <-- I/O BUFFER FOR PORT # 7
15405   (   104) <-- I/O BUFFER FOR PORT # 6
15511   (   104) <-- I/O BUFFER FOR PORT # 5
15615   (  1015) <-- $MMUX
16630   (  2277) <-- $CTUS
21131   (   237) <-- $TRMTV950
21370   (  1305) <-- $LCM
22675   (   275) <-- $MTAO
23172   (   101) <-- $CALLTBL
23273   (   220) <-- RUN'S CALLTABLE
23523   (   270) <-- $TERMTV912
24015   (    45) <-- I/O BUFFER FOR PORT # 4
24062   (    37) <-- INTERRUPT STACK
24121   (   104) <-- I/O BUFFER FOR PORT # 3
24225   (   104) <-- I/O BUFFER FOR PORT # 2
24331   (   154) <-- $TERMS
24505   (   204) <-- TERMINAL TYPE TABLE (PART OF $TERMS)



24741   (   620) <-- $SYS SCHED
25561   (   177) <-- DISCSUB # 27
25740   (   760) <-- DISCSUB # 61
26740   (    15) <-- LUVAR FOR LU # 0
26755   (  1023) <-- LUFIX FOR LU # 0,
30000   (   400) <-- BSA (BLOCK SWAP AREA)
30400   (   400) <-- HBA (HEADER BLOCK AREA)
31000   (   400) <-- HXA (HEADER EXTENDER AREA)
31400   (   400) <-- SSA (SUBROUTINE SWAP AREA)
32000   (   200) <-- 4 FREE NODES
32200   (11000) <-- BPS (BEGINNING OF PROCESSOR STORAGE)
43200   (     2) <-- LEPS (LOCATION OF END OF PROCESSOR STORAGE)
43202   (  1700) <-- 30 FREE NODES
45102   (  5000) <-- POOL BUFFER # 1 THROUGH 10
52102   (  1271) <-- BUFFER POOL TABLE
53373   (    50) <-- INTERMEDIATE INPUT BUFFER FOR PORT # 36
53443   (    50) <-- INTERMEDIATE INPUT BUFFER FOR PORT # 35
53513   (    50) <-- INTERMEDIATE INPUT BUFFER FOR PORT # 34
53563   (    50) <-- INTERMEDIATE INPUT BUFFER FOR PORT # 33
53633   (    50) <-- INTERMEDIATE INPUT BUFFER FOR PORT # 32
53703   (    50) <-- INTERMEDIATE INPUT BUFFER FOR PORT # 31
```

Figure 2-4. Partial COREMAP Printout

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 IRIS system 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 IRIS system 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 Displays the end of the file; the number of the last line is then displayed

```
### 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 IRIS system 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)
```

DSP

DSP (Disk Service Processor) is a machine language editor that can be used to edit memory and disk locations. For more information, see the IRIS R9 System Manager Manual.

EDIT

EDIT is used to create and edit text files. For more information, see Section 3, IRIS Editors.

EXERCISER

EXERCISER is a program that tests the ability of the disk controller or Lotus Cache Memory to do single block transfers from formatted files, and it tests the ability of the memory buffer pool to maintain that information.

When a system is installed, EXERCISER is used in conjunction with SWAPTEST to test the complete interaction of the IRIS Operating System, the computer, and the disk. POINT 4 recommends that EXERCISER and SWAPTEST be run overnight and concurrently, each test on several terminals. If no errors or malfunctions occur, these tests will run forever unless aborted by an operator. If an error or a malfunction occurs in a test, that test aborts and the IRIS system prompt (#) is displayed.

To initiate EXERCISER on one or more terminals, enter the following at the IRIS system prompt (#) on each of the selected terminal keyboards:

```
EXERCISER
```

The following is displayed:

```
#EXERCISER

THIS CORE AND DISC EXERCISER PROGRAM WAS NOT DESIGNED TO BE A
REPLACEMENT FOR A COMPREHENSIVE STAND-ALONE RELIABILITY PROGRAM.
RATHER, IT IS A CONVENIENT TOOL WHICH CAN BE RUN USING LIVE
DATA PACKS, WITHOUT HAVING TO ASK EVERYONE ELSE TO LOG OFF.
POINT 4 ADIVSES YOU TO RUN IT OVERNIGHT OR WEEKENDS.  ANY ERROR
FOUND IN CORE OR ON DISC WILL ABORT THE PROGRAM AND PRINT
AN ERROR MESSAGE.

CHOOSE THE # OF BLOCKS TO BE USED BY THIS EXERCISER.
THE # SHOULD BE GREATER THAN THE # OF BLOCKS IN THE BUFFER
POOL, IF POSSIBLE.
CONVERT THE # OF BLOCKS FROM OCTAL TO DECIMAL.
MAKE SURE THAT THE CHOSEN BLOCK COUNT DOES NOT EXCEED THE NUMBER OF
BLOCKS AVAILABLE TO THIS ACCOUNT ON THE SELECTED LOGICAL UNIT.
ENTER # OF BLOCKS TO USE:
```

Enter the number of blocks to use in accordance with the message. A minimum of 1000 blocks is recommended, although fewer may be entered if 1000 blocks are not available. The following is displayed:

```
ENTER LOGICAL UNIT # TO USE:
```

Enter the number of the appropriate logical unit.

The following example uses a block count of 60 so that a complete test can be shown:

```
HAVE COMPLETED THROUGH BLOCK  10 OR WRITE OUT PHASE
HAVE COMPLETED THROUGH BLOCK  20 OF WRITE OUT PHASE
HAVE COMPLETED THROUGH BLOCK  30 OF WRITE OUT PHASE
HAVE COMPLETED THROUGH BLOCK  40 OF WRITE OUT PHASE
HAVE COMPLETED THROUGH BLOCK  50 OF WRITE OUT PHASE
HAVE COMPLETED THROUGH BLOCK  60 OF WRITE OUT PHASE
HAVE COMPLETED THROUGH BLOCK  10 OF READ BACK AND VERIFY PHASE
HAVE COMPLETED THROUGH BLOCK  20 OF READ BACK AND VERIFY PHASE
HAVE COMPLETED THROUGH BLOCK  30 OF READ BACK AND VERIFY PHASE
HAVE COMPLETED THROUGH BLOCK  40 OF READ BACK AND VERIFY PHASE
HAVE COMPLETED THROUGH BLOCK  50 OF READ BACK AND VERIFY PHASE
HAVE COMPLETED THROUGH BLOCK  60 OF READ BACK AND VERIFY PHASE

POINT 4 CORE AND DISC EXERCISER (VERSION n)
WITH # OF BLOCKS IN TEST = 60
PASS 3          n              COMPLETED OK. (NO ERRORS)
PRESS CONTROL C TO ABORT
```

To end the test, press <CTRL-C>. The following is displayed:

```
USER REQUESTED EXIT
#
```

After the test is completed, run PORT ALL MONITOR to check whether all test ports are still in RUN. If any are at SCOPE (the IRIS system prompt), it indicates a hardware malfunction has occurred, and hardware diagnostics should be run.

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 IRIS 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 $TRXCO and EXTRAPORT. A CALL $TRXCO may be inserted into a BASIC program. Please refer to the IRIS R9 Business BASIC Manual for information on this procedure.

EXTRAPORT uses CALL $TRXCO to initiate a job.

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

To run on a phantom port, at the IRIS system 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 IRIS system manager.

FAULTPRINT

This utility is used by POINT 4 software to display TRAP messages.

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 IRIS system prompt (#), enter

```
FINDFILE
```

The following message is then displayed:

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

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 LU 1  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!!!
*****************

ALL DONE!!!

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

FORGE

FORGE is used to create and edit text files. For more information, see Section 3, IRIS Editors.

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.

Formatted Data Files

A formatted file consists of a file header and up to 32,768 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.

The maximum capacity of a formatted file is 65,535 records or 32,768 disk blocks, whichever is less. Thus, the maximum file size is dependent on the record size chosen. For example, the smallest record size of one word per record allows a maximum file size of 65,535 records because of the record limit. (This is equivalent to 256 blocks.) The largest record size of 256 words (one disk block) per record allows a maximum file size of 32,768 records because of the disk block limit.

CREATING A FORMATTED FILE

To create a new formatted file, at the IRIS system 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 file by other users
lu - number of logical unit where 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:

The following 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: o D or D1 specifies a one-word decimal integer in the range ±7999). o 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 for item.

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

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 unused words in each block). Assume the following requirements:

Item	Description
#0	A string of 20 bytes
#1 thru #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 R9 Business BASIC Manual for information on the procedure.

Contiguous Data Files

A contiguous file consists of a header and a number of data blocks in a contiguous area on the disk. These contiguous blocks must be allocated when the file is first built, whether all blocks will be used immediately or not.

Contiguous files will accommodate mixed record formats.

CREATING A CONTIGUOUS FILE

To create a contiguous file, at the IRIS system 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 65535 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/ - logical unit on which file is to be built

! - permits replacement of an existing contiguous file

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 a contiguous file of equal or greater size (deleting a formatted or text file may not create sufficient contiguous space); the space does not become available until the channel is closed
- Running the utility CLEANUP to reorganize the logical unit
- Using INSTALL AND CLEAR to create a new logical unit; this deletes all files and accounting information on the LU

EXAMPLE OF CREATING 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
```

GUARD

Several BASIC functions are restricted to use on the IRIS manager account. These functions are desirable and useful when handled in a cautious and knowledgeable manner, but because they are very powerful, they are also very dangerous and could cause serious problems to the operating system and user data and programs if misused.

The IRIS system manager may want to allow another user to run a program containing one of the restricted functions, but does not want to give that user the manager password. For example, a junior operator does backups late at night or early in the morning. The manager wants this junior operator, when bringing up the system, to set the date and time from a BASIC program that checks the input for reasonableness. The manager wants to avoid giving out the password or making CALL $TIME available from any account. The GUARD utility provides the solution to this dilemma. The manager can write a BASIC program using CALL $TIME, and then GUARD it to allow it to run from any account; once GUARDed, that program cannot be modified or listed.

GUARD Features

The GUARD utility program provides a method by which the IRIS system manager can allow limited access to restricted functions. These restricted functions include:

- Open File Maintenance - allows the header and data of any file to be examined and modified (similar to DSP)
- CALL 95 - allows reading or writing any numeric or string from or to any logical unit
- CALL $RDFHD - allows reading of file headers
- CALL $SINGLE - sets IRIS into or out of single-user mode
- CALL $TIME - allows the system clock and date to be changed
- SPC (32768 + N) - allows any address N in lower memory (i.e., below 32KW) to be read
- SPC (65536 + N) - allows any address N in lower or upper memory to be read

These functions are all available from the IRIS manager account without being GUARDed.

Before GUARDing a program, the manager can add any desired level of checking by using SPC 5 or SPC 6 to restrict usage or by checking the desired date and time. (For an explanation of SPC functions, refer to the IRIS R9 Business BASIC Manual.)

The header of each BASIC program contains a set of bits called the Designated Operator Oriented Mode (DOOM) bits. These bits can be set by specifying the appropriate option in GUARD as described in Table 2-3. The bits can only be set (i.e., enabled or disabled) from the IRIS manager or utility accounts; when a program is first created, all the bits in the GUARD word are disabled. Once any GUARD bit has been set, the program is said to be GUARDed.

GUARD option 5 prevents both listing and modification but is not permanent because the option can be reset. To permanently protect a program, see PROTECT.

If a program is GUARDed but option 5 is disabled, the program mode is "execute only" unless the program is run from the IRIS manager account.

SECURITY OF GUARDED PROGRAMS

If a program has been granted open file maintenance capability, the program is execute-only to all users except the IRIS system manager. If any user other than the manager attempts to load the program using BASIC, BASIC immediately clears the program from the user's partition as though "NEW" had been entered. The program still resides on disk, so it is safe.

To provide additional security, option 5 of GUARD can be used to set a bit to make a program execute-only to everyone, including the manager. The system checks this bit when the manager attempts to bring any program into his or her active file. This bit should be set to protect any program used by the IRIS system manager to set the bits of other programs. Any program used to set the bits of other programs should require entry of a password in order to prevent any user who happens to be on the IRIS manager account from setting the bits of programs without the manager's knowledge or consent.

TABLE 2-3. GUARD OPTIONS

Option	Description
1	READ-WRITE FILES OBEYING FILE PROTECTION - Allows use of open file maintenance to access files regardless of file type only if not prevented by the account's privilege or the file's protection levels
2	READ-WRITE FILES IGNORING FILE PROTECTION - Allows use of open file maintenance to access files regardless of file type, privilege and protection levels
3	WRITE TO MEMORY - Allows use of CALL $TIME to set system time, or CALL $SINGLE to set IRIS into or out of single-user mode; also allows use of CALL 93
4	READ FROM MEMORY - Allows use of SPC (65536 + N) or SPC (32768 + N) for unrestricted reading of memory
5	EXECUTE ONLY - CAN NOT BE LISTED - Prevents listing or modifying a BASIC program by anyone including the IRIS manager
6	READ-WRITE FROM DISK - Allows use of CALL 95 by other than the IRIS manager account
7	READ FILE HEADER (CALL $RDFHD) - Allows use of CALL $RDFHD by other than the IRIS manager account

Using the GUARD Program

The GUARD program is an interactive program displaying appropriate prompts and messages. The user may exit the program at any time by pressing <ESC> or <CTRL-C>.

The procedure for using GUARD is as follows:

1. Log on to the IRIS manager account.

2. At the IRIS system prompt (#), enter

   #GUARD

   The program then displays

   GUARD Version n.nn

   where
   n.nn - current revision number

   The following prompt is then displayed:

   ENTER PASSWORD:

3. Enter the GUARD program's password (the default is X; for information on changing passwords, see the IRIS R9 System Manager Manual).

   The password is not echoed. If an incorrect password is entered, the program aborts and the IRIS system prompt is displayed.

   If the correct password is entered, GUARD requests the name of the BASIC program to be GUARDed:

   BASIC PROGRAM NAME

4. Enter the name of the BASIC program. To specify a BASIC program residing on another logical unit, enter the name in the form

   lu/programname

   If the filename entered cannot be found or is not a BASIC program, an appropriate message is displayed and the program name prompt is repeated.

If a valid BASIC program name is entered, GUARD displays the current settings of the GUARD bits as shown in the following example (user input is underlined):

```
BASIC PROGRAM NAME 6/TESTER

                THE GUARD WORD FOR    6/TESTER    IS NOW:

1 .....READ-WRITE FILES OBEYING FILE PROTECTION          DISABLED

2 .....READ-WRITE FILES IGNORING FILE PROTECTION         DISABLED

3 .....WRITE TO MEMORY                                   DISABLED

4 .....READ FROM MEMORY                                  DISABLED

5 .....EXECUTE ONLY - CAN NOT BE LISTED                  DISABLED

6 .....READ-WRITE DISK                                   DISABLED

7 .....READ FILE HEADER (CALL $RDFHD)                    DISABLED

OPTION NUMBERS TO CHANGE (#,#) :
```

5. Enter the number of the option to be enabled. If more than one option is to be enabled, the desired option numbers may be entered at one time provided they are separated by commas.

   If the option numbers are not separated by commas, the numbers are rejected as shown in the following example:

```
OPTION NUMBERS TO CHANGE (#,#) : 23

23 .....IS NOT A VALID OPTION NUMBER
```

The prompt, OPTION NUMBERS TO CHANGE, is redisplayed.

If the option numbers are entered correctly, the user is asked to confirm each option chosen as shown in the following example:

```
OPTION NUMBERS TO CHANGE (#,#) : 1,3

1 .....READ-WRITE FILES OBEYING FILE PROTECTION  IS NOW DISABLED

                                          ENABLE IT (Y OR N) ? Y

3 .....WRITE TO MEMORY    IS NOW DISABLED

                                          ENABLE IT (Y OR N) ? Y

                THE GUARD WORD FOR    6/TESTER    IS NOW:

1 .....READ-WRITE FILES OBEYING FILE PROTECTION    ENABLED

2 .....READ-WRITE FILES IGNORING FILE PROTECTION          DISABLED

3 .....WRITE TO MEMORY                             ENABLED

4 .....READ FROM MEMORY                                   DISABLED

5 .....EXECUTE ONLY - CAN NOT BE LISTED                   DISABLED

6 .....READ-WRITE DISK                                    DISABLED

7 .....READ FILE HEADER (CALL $RDFHD)                     DISABLED

                                      IS THIS CORRECT (Y OR N) ? Y

THE GUARD WORD HAS BEEN WRITTEN FOR 6/TESTER


BASIC PROGRAM NAME
```

6. Enter the name of the next program to be GUARDed and continue the procedure, or press <RETURN> to exit the GUARD utility.

GUIDE

GUIDE is used to obtain parameters needed to use the stand-alone program BLOCKCOPY. For more information, see the IRIS R9 System Manager Manual.

INSTALL

INSTALL is the system command that assigns logical units to disk partitions. INSTALL has an option word that can be used to specify the accounts from which INSTALL can be run. For more information on the option word, see the IRIS R9 System Manager Manual.

INSTALL can be used to

- Structure logical units on a new or existing disk partition
- Install all logical units
- Install a specific logical unit
- Change a logical unit number
- Report bad blocks on a nonzero logical unit

INSTALL is also used in the procedures to accomplish the following:

- Enable the transfer of nonzero logical units
- Change the size of an existing logical unit
- Access a damaged logical unit for repair and/or file salvaging

As part of its operation, INSTALL constructs the DMAP (except in INSTALL FAST), checks the integrity of the file structures, and adds the logical unit numbers to the LU table. It also updates LUVAR and LUFIX (internal software tables) and checks the logical unit base year against the system base year.

Restrictions regarding the use of INSTALL include the following:

- LU 0 and LU 5 contain the IRIS system software and application programs and should not be changed in any way.
- The maximum size for an LU is 65,536 blocks.

The following subsections describe the various uses of INSTALL, and provide lists of the messages and error codes associated with INSTALL.

Structuring Logical Units (LUs)

Logical units can be structured on new or existing disk partitions. Structuring LUs on new partitions occurs when the system is first set up or when additional partitions are added to the Disk Driver Table.

A logical unit can be structured on an existing disk partition provided that the area on disk is not contained in any current LU. Changing the size of an LU is more complex, however; and the methods for increasing or decreasing size are described in the section, Changing the Size of an Existing Logical Unit.

The following subsections describe the more common methods for structuring logical units on new and existing partitions.

CREATING A NEW PARTITION

Before a logical unit can be structured and its number assigned to a new disk partition, SETUP must be used to create the disk partition as follows:

1. Back up the system.
2. Enter SETUP and define the new partition in the Disk Driver Table.
3. If necessary, change the system maximum number of installed LUs in the System INFO Table.
4. Run the update option on SETUP.

For more information on SETUP, refer to the IRIS R9 System Configuration Manual.

STRUCTURING A NEW LOGICAL UNIT

After the disk partition has been created, a logical unit can be structured on the new disk partition as follows:

1. Log on to the manager or designated account.

2. Enter the following at the IRIS system prompt (#):

```
INSTALL d.p
```

where
d - number of logical controller
p - number of partition

For example, INSTALL 0.2 refers to logical controller 0 and partition number 2.

INSTALL looks at block 1 of the designated disk partition to determine whether the partition has been previously structured as an IRIS LU. If the system finds that the partition has not been previously structured as an IRIS LU, the following is displayed:

```
! NOT AN IRIS LOGICAL UNIT . . .
DESIRED LOGICAL UNIT NUMBER ?
```

3. Enter the desired logical unit number. The following is displayed:

```
BAD BLOCK ?
```

Any known bad block should be reported to prevent IRIS from using it. If there is a bad block to report, see the section, Reporting Bad Blocks During an INSTALL. If there is none to report, press <RETURN>. The following is displayed:

```
LOGICAL UNIT #n IS NOW ACTIVE !

#
```

STRUCTURING AN LU ON AN EXISTING PARTITION

A new logical unit can be installed on an existing disk partition that is structured for IRIS LUs if that area is not already contained in a current LU. The process clears the disk partition (i.e., clears and kills all existing files and account information), requests a new number, and structures a new, empty LU.

To clear the disk partition and structure a new LU, enter the following at the IRIS system prompt (#):

```
INSTALL AND CLEAR d.p
```

where
- d - number of logical controller
- p - number of partition

For example, INSTALL AND CLEAR 0.2 refers to logical controller 0 and partition number 2.

INSTALL looks at block 1 of the designated disk partition to determine whether the disk partition has been previously structured as an IRIS LU. If it has been previously structured as an IRIS LU, INSTALL requests confirmation that this logical unit is to be cleared as follows:

```
LOGICAL UNIT NUMBER = n
CLEAR (Y/N) ?
```

If the logical unit is to be installed or changed, rather than cleared, enter N and refer to the appropriate subsection that follows. To clear the logical unit, enter Y. The following is displayed:

```
DESIRED LOGICAL UNIT NUMBER ?
```

Enter the desired number for the logical unit. The following is displayed:

```
BAD BLOCK ?
```

Any known bad block should be reported to prevent IRIS from using it. If there is a bad block to report, see the section, Reporting Bad Blocks During an INSTALL. If there is none to report, press <RETURN>. The following is displayed:

```
LOGICAL UNIT #n IS NOW ACTIVE !

#
```

Installing Logical Units

INSTALL can be used to install all logical units or a specific one. It can also be used to change a logical unit number. The following subsections describe these functions of INSTALL; a procedure for reporting bad blocks during INSTALL is also provided.

INSTALLING ALL LUs

To install all previously structured logical units, enter the following at the IRIS system prompt (#):

```
INSTALL
```

The following is displayed:

```
PLEASE WAIT . . .
```

Logical units are installed in the order that they appear in the Disk Driver Table of SETUP. For each logical unit, INSTALL checks the integrity of each file's structure. If a bad file is found, an appropriate message is displayed, installation is terminated and control is returned to the IRIS system prompt (#). If none is found, the following is displayed:

```
BAD BLOCK ?
```

Any known bad block should be reported to prevent IRIS from using it. If there is a bad block to report, see the section, Reporting Bad Blocks During an INSTALL. If there is none to report, press <RETURN>. The following is displayed:

```
LOGICAL UNIT #n IS NOW ACTIVE !

PLEASE WAIT . . .
BAD BLOCK ?
```

This prompt is repeated for each logical unit. If there is a bad block to report, see the section on reporting bad blocks; otherwise press <RETURN>.

When the last logical unit has been installed, the IRIS system prompt (#) is displayed as follows:

```
LOGICAL UNIT #n IS NOW ACTIVE !

#
```

INSTALLING A SPECIFIC LU

If a specific logical unit is to be installed, enter the following at the IRIS system prompt (#):

```
INSTALL d.p
```

where
d - number of logical controller
p - number of partition

For example, INSTALL 0.2 refers to logical controller 0 and partition number 2.

INSTALL looks at block 1 of the designated disk partition to determine whether the disk partition has been previously structured as an IRIS LU. If it has, the following message is displayed:

```
#INSTALL d.p
LOGICAL UNIT NUMBER = n
INSTALL (Y/N) ?
```

If the logical unit number is to be changed, enter N and refer to the next section, Changing a Logical Unit Number. To install the logical unit with the number shown, enter Y.

If the partition has not been installed, INSTALL checks for disk map (DMAP) errors. If an error is found, an appropriate message is displayed (see the section, Errors During an INSTALL); otherwise, the following is displayed:

```
PLEASE WAIT . . .
```

INSTALL checks the integrity of each file's structure. If a bad file is found, an appropriate message is displayed, installation is terminated and control is returned to the IRIS system prompt (#). If none is found, the following is displayed:

```
BAD BLOCK ?
```

Any known bad block should be reported to prevent IRIS from using it. If there is a bad block to report, see the section, Reporting Bad Blocks During an INSTALL. If there is none to report, press <RETURN>. The following is displayed:

```
LOGICAL UNIT #n IS NOW ACTIVE !

#
```

To install other specific logical units, repeat this process from the beginning.

Changing an LU Number

Before changing the number of a logical unit, obtain the list of disk drives and partitions configured for the system from the IRIS system manager. The range of permissible LU numbers is 1 through 127. Once an LU is changed, all subsequent references to the LU must be by the new number. LU 0 is the system logical unit; it cannot be changed!

Log on to the manager or designated account and enter

```
INSTALL d.p
```

where
- d - number of logical controller
- p - number of partition

For example, INSTALL 0.2 refers to logical controller 0 and partition number 2.

INSTALL looks at block 1 of the designated disk partition to determine whether the disk partition has been previously structured as an IRIS logical unit. If it has, the following message is displayed:

```
#INSTALL d.p
LOGICAL UNIT NUMBER = n
INSTALL (Y/N) ?
```

If Y is entered, the logical unit is installed with the number shown. To change the logical unit number from that shown, enter N. The following is displayed:

```
CHANGE LOGICAL UNIT # (Y/N) ?
```

If N is entered, the installation is terminated. To change the logical unit number, enter Y. The following is displayed:

```
NEW LOGICAL UNIT NUMBER ?
```

Enter the desired number and press <RETURN>. The following is displayed:

```
PLEASE WAIT . . .
```

INSTALL checks the integrity of each file's structures. If a bad file is found, an appropriate message is displayed, installation is terminated, and the IRIS system prompt (#) is displayed. If none is found, the following is displayed:

```
BAD BLOCK ?
```

Any known bad block should be reported to prevent IRIS from using it. If there is a bad block to report, see the section, Reporting Bad Blocks During an INSTALL. If there is none to report, press <RETURN>. The following is displayed:

```
LOGICAL UNIT #n IS NOW ACTIVE !

#
```

Reporting Bad Blocks for Nonzero LUs During an INSTALL

NOTE

> Reporting bad blocks during an INSTALL or IPL is a temporary solution. The recommended procedure for handling bad blocks is to reformat the disk and then perform a Restore using DISCUTILITY.

A bad block is an area on the disk that cannot be accessed reliably. When the system encounters a bad block, it displays a Trap 3 or 5 message on the terminal that was accessing the disk. The information in the trap message should be recorded and corrective action should be taken.

The trap message contains the real disk address (RDA) of the bad block, which is given in Register A1. For a nonzero logical unit, the RDA should be entered during INSTALL to ensure that the system does not try to use it. A bad block has to be reported only once for nonzero LUs. Note that the recommended solution is to reformat the disk.

No opportunity is provided to enter bad blocks for LU 0 during an INSTALL. (To enter bad blocks for LU 0, see Initial Program Load in the IRIS R9 System Manager Manual.)

To report bad blocks on a specific LU during an INSTALL, follow these steps:

1. In response to the BAD BLOCK ? prompt, enter

   ```
   nn
   ```

   where
   nn - the real disk address (RDA) of the bad block reported in Register A1 of the trap message

2. Press <RETURN>. The prompt repeats

   ```
   BAD BLOCK ?
   ```

3. Enter the real disk address (RDA) of the next bad block. Repeat this procedure until all bad blocks have been reported.

4. After the last bad block has been reported, press <RETURN> at the BAD BLOCK ? prompt. The following message is displayed:

   ```
   LOGICAL UNIT #n IS NOW ACTIVE !

   #
   ```

Transferring Nonzero Logical Units (LUs)

Nonzero logical units may be transferred from one system to another and from one type of drive to another using a variety of output media and software. INSTALL helps in the transfer process. Valid media and associated utility programs used in the transfer process are listed in Table 2-4.

To transfer a nonzero logical unit, follow these steps:

1. INSTALL the specific logical unit that is to be transferred.

2. Run CLEANUPX to compress the files and to create a special area for the DMAP.

3. SHUTDOWN immediately to the appropriate transfer utility program.

4. Transfer the logical unit to the desired destination disk using any of the media and associated utility programs shown in Table 2-4.

5. INSTALL the transferred logical unit using the procedure for installing a specific logical unit. The procedure is the same except for the following message, which is displayed before the PLEASE WAIT message:

   DMAP ADJUSTMENT IN PROGRESS

   If CTUTILITY is used to transfer the logical unit, this message is not displayed.

6. Run CLEANUP on the transferred logical unit and the original logical unit. This reorganizes the logical unit index and improves system performance.

TABLE 2-4. MEDIA FOR TRANSFERRING A NONZERO LOGICAL UNIT

Media	Software
Cassette Tape*	CTUTILITY (see IRIS R9 System Manager Manual)
1/4 or 1/2-inch	DISCUTILITY Save, Verify, and Restore options (see appropriate DISCUTILITY manual)
Disk	LOTUS or MARK 3 DISCUTILITY Convert option; BLOCKCOPY* (see IRIS R9 System Manager Manual)

*Allows transferring between unlike disk drives.

Changing the Size of an Existing Logical Unit (LU)

Existing logical units can be increased and decreased in size using the instructions in the following subsections.

INCREASING THE SIZE OF AN EXISTING LU

A logical unit containing programs or data can be increased in size if additional cylinders are available on the disk following that LU, or if other LUs are decreased in size to make room. To decrease an LU, see the next section, Decreasing the Size of a Logical Unit.

To increase the size of an existing logical unit, perform the following steps:

1. Back up the system, one LU at a time, using a separate backup tape for each LU.

2. Run a LIBR lu/? to determine the number of blocks available on the LU to be increased. This is necessary because step 3 requires that 152 (decimal) blocks be available on the LU as work space.

3. If the LIBR lu/? indicates that 152 or more blocks are available, run CLEANUPX (not CLEANUP) on the LU. If less than 152 blocks are available, transfer some files to another LU, then run CLEANUPX. The transferred files can be returned to the LU after its size has been increased.

4. Enter SETUP and modify the Disk Driver Table to increase the number of cylinders for the LU. Then, decrease the cylinders for other LUs, if necessary, to accommodate this LU.

5. Run the update option of SETUP.

6. INSTALL the logical unit.

DECREASING THE SIZE OF AN EXISTING LU

It may be desirable to decrease the size of an existing logical unit that has an excess of unused blocks in order to accommodate an increased LU, or to create two smaller LUs in place of a larger one. Caution should be exercised when decreasing the size of a logical unit to avoid losing files or parts of files from that logical unit.

There are two methods for decreasing the size of a logical unit. The first assumes that all files on the LU to be decreased can be copied onto a second LU while the first is being decreased. The second decreases the size of the LU while the files are still resident. The second method is risky and should be done only by an advanced manager.

To decrease the size of a logical unit by copying its files to another LU, follow these steps:

1. Use the IRIS COPY command to copy all the files on the LU to be decreased to another LU.

2. Enter SETUP and modify the Disk Driver Table to decrease the number of cylinders for the LU.

3. Run the update option of SETUP.

4. Use INSTALL AND CLEAR to clear the LU to be decreased and to structure the new LU.

5. Use COPY to copy the files back to the smaller LU.

To decrease the size of a logical unit while the files are still resident on the LU, advanced managers can use these steps:

1. Back up the system.

2. Run LIBR lu/? to determine the number of blocks available on the LU to be decreased. This is necessary before implementing step 3, which requires that 152 (decimal) blocks be available on the LU as work space.

3. If the LIBR lu/? indicates that 152 or more blocks are available, run CLEANUP (not CLEANUPX).

4. INSTALL the logical unit.

5. Run LIBR lu/? again and note the number of available blocks. Reduce this number by 10% to allow for work space, then convert this number to octal.

6. Refer to the appropriate Disk Specification sheet in the IRIS R9 Peripherals Handbook for the LRC (the number of blocks per cylinder in octal). Divide the result obtained in Step 5 by the LRC. This determines the maximum number of cylinders that can be removed from the LU.

7. Modify the Disk Driver Table in SETUP to reduce the number of cylinders in that LU. Do not reduce it more than the limit calculated in step 6. Change the starting cylinder numbers of the LUs that follow the decreased LU.

8. Run the update option of SETUP.

9. ReINSTALL the logical unit.

CAUTION

These steps must be followed very carefully. If the LU is reduced by too many cylinders, a Trap 5 may occur when accessing the files, which indicates that some files are no longer intact. If this occurs, it is necessary to go to the backup copy, recheck the calculations, and repeat the procedure.

Accessing a Damaged Logical Unit (LU) for Repair and File Salvaging

INSTALL FAST is a system command that can be used by the manager or other designated account to install a logical unit without rebuilding the Disk Map (DMAP). It is used as a last resort when INSTALL refuses to install. INSTALL FAST is used only to access a damaged LU for repair or breakdown operations, or to attempt to salvage files from a damaged LU.

After INSTALL FAST has been used, the LU is in a vulnerable condition. The more the LU is used, the greater the danger of spreading damage over the entire LU. Unless absolutely necessary, use a regular INSTALL which builds a complete DMAP.

WARNING!

> Do not use INSTALL FAST after a system crash or if the data on the logical unit being installed is suspect, as damage to the data may occur.

To use INSTALL FAST, enter the following at the IRIS system prompt (#):

```
INSTALL FAST d.p
```

where
- d - number of logical controller
- p - number of partition

For example, INSTALL 0.2 refers to logical controller 0 and partition number 2.

SALVAGING QUESTIONABLE OR DAMAGED FILES

If a questionable or damaged file is encountered during an INSTALL, it is handled according to the parameters set at location 204 (octal):

- If the bit is set to 0, INSTALL retains the file and terminates; control returns to the IRIS system prompt (#). After the file condition is resolved, INSTALL can be repeated, but INSTALL will terminate again at any subsequent bad file. The following message is displayed:

```
BAD FILE ENCOUNTERED
FILE NAME xxxxx
INDEX BLOCK # rrr
DISPLACEMENT nnn
ERROR CODE c
INSTALLATION TERMINATED
```

  where
  xxxxx - name of bad file in INDEX
  rrr - RDA of INDEX block containing bad file
  nnn - displacement in that INDEX block
  c - error code (see Table 2-5)

  If a block of the INDEX has been damaged, the screen may be unintelligible or the file name meaningless.

- If the bit is set to 1, INSTALL retains a file that is being built but deletes a damaged one.

- If the bit is set to 2, INSTALL deletes a file that is being built or is damaged.

The most secure way to recover a questionable or damaged file is to recover the file from a good backup and to reenter the missing data. However, the following steps may also be used to salvage questionable or damaged files:

1. Ensure that all users are logged off.

2. Back up the system.

3. Do an INSTALL FAST.

4. Use COPY to copy the bad file to a good LU.

5. If COPY fails, write a small BASIC program to read data out of the bad file and write it to a file on another LU.

6. If both steps 4 and 5 fail, restore the file from a backup and reenter missing data.

7. REMOVE the suspect LU.

Even if either step 4 and 5 is partly or wholly successful, there is still the possibility that some of the data is bad. A knowledgeable applications programmer may be able to write a program that checks the file record by record for damaged data.

TABLE 2-5. INSTALL ERROR CODES

Error Code	Description
1	File being built or deleted
2	File restrictions violated
3	Filename at offset NAME (see DEFS listing, IRIS R9 System Configuration Manual) in header block does not match entry in the INDEX
4	Real disk address at offset DHDR in the file's header does not match entry in the INDEX
5	The RDA list in the file's header block does not contain enough entries as indicated by NBLK
6	File is attempting to claim a block which is already marked in DMAP as in use by a previous file
7	Reserved
10	The RDA list in the file header block contains too many entries as indicated by NBLK

Messages That Can Occur with INSTALL

The following messages may occur when entering commands during INSTALL:

?PARTITION IS ALREADY INSTALLED INSTALLATION TERMINATED. #	The partition has already been structured as an IRIS LU. Check the Disk Driver Table; install another partition if necessary.
?ILLEGAL PARTITION NUMBER INSTALLATION TERMINATED. #	An illegal partition number has been attempted. Check the Disk Driver Table.
LOGICAL UNIT NUMBER IS IN USE ! DESIRED LOGICAL UNIT NUMBER ?	The number entered is already in use. Enter the number of an LU not in use.
LOGICAL UNIT NUMBER IS IN USE ! CHANGE LOGICAL UNIT # (Y/N) ?	The number entered is already in use. Enter the number of an LU not in use or change the LU number.
?ILLEGAL COMMAND INSTALLATION TERMINATED.	Command was entered incorrectly. Reenter the command.
CAUTION: THIS LOGICAL UNIT SHOULD BE REHASHED BEFORE USE	May occur when a LU is transported from one operating system to another. See REHASH.

Errors During an INSTALL

DMAP (disk map) is a system file on each logical unit. It records which real disk addresses (RDAs) are in use for files on that logical unit. If INSTALL finds that DMAP is not the proper DMAP for the logical unit, one of the following messages is displayed:

```
DMAP HEADER NOT IN HBA
NCYL IS TOO LARGE
NTRS OR DHDR IS INVALID
NTRS OR NRPB IS INVALID
```

where

NCYL - total number of cylinders
NTRS - number of IRIS tracks/sectors for LU to be INSTALLed
DHDR - real disk address of the DMAP file header
NRPB - number of records per block in DMAP header

DMAP Header Not in HBA - indicates that the real disk address for the DMAP INDEX entry was not at IRIS track 1, sector 0. This may occur if an LU was ported to a different type of drive without first running CLEANUPX.

NCYL is Too Large - can occur if a user attempts to INSTALL an LU at a larger size than it was previously, but did not run CLEANUPX before INSTALLing at the new size.

The other messages may indicate that an LU was ported to a different type of drive without running CLEANUPX, or that a possible discrepancy exists between the DMAP header and the partition's attributes as defined in the Disk Driver Table in the IRIS R9 System Configuration Manual. Refer to the appropriate Disk Specification sheet in the IRIS R9 Peripherals Handbook for the correct values and compare them to the entries in the Disk Driver Table.

KILL

One or more 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 that is not write-protected, may be deleted.

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

Using KILL

To delete one file or program, at the IRIS system prompt (#), 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 on which the file resides

If the deletion is successful, the message displayed is

```
DELETED
```

If the deletion is unsuccessful, an appropriate message is displayed (see messages in next subsection, Example of Using KILL).

More than one file (which may reside on different LUs) may be deleted at a 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 deletion is successful, the program displays

```
ALL DELETED
```

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. The third file (4/MAX) is on another user's account and is write-protected. The sixth file (COPY) is a system file that cannot be deleted without first changing it to another file type. The seventh file (2/PAY) is on an LU that is not installed.

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

      1     2  3     4  5  6     7
```

In this example, 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.

Deleting Polyfile Volumes

Polyfiles need special handling and it is recommended that KILLPF (see KILLPF) be used. However, if KILL is used, volumes should be deleted one at a time, ending with volume 0.

To delete a polyfile volume using KILL, at the IRIS system prompt (#), 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.

KILLPF

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

At the IRIS system 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:

```
#KILLPF
 KILLPF - Kill Polyfile Utility
 Polyfile name [should have "LU/"]: 1/pfname  Validating polyfile structure.
 Please wait...
 March 16, 1986  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
```

An asterisk following a volume number indicates that there is an error in the polyfile structure. For an explanation of possible errors, see the section, QUERYPF, Polyfile Global Display.

After YES is entered, each volume number is displayed 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.

LCMACTIVATE

LCMACTIVATE is a system command that activates both extended memory and Lotus Cache Memory drivers. LCMACTIVATE is normally executed as part of the system manager log-on procedure after the installation of logical units.

It is useful to remember that a MARK 12 can have either extended memory or LCM but not both, and that LCMACTIVATE is used to activate either type of memory. The MARK 5 and 8 can have only LCM. The MARK 9 can have LCM and mapped memory. If both LCM and mapped memory are present in a MARK 9, each must be activated separately.

To activate extended memory or LCM, enter the following at the IRIS system prompt (#):

```
LCMACTIVATE
```

When the extended memory or LCM is activated, the following is displayed:

```
LCM IS NOT ACTIVE
```

LCMCHECK

LCMCHECK is a BASIC program that is used to examine the current status of the LCM driver. The output includes all information set by LCMACTIVATE plus any statistical information accumulated by the LCM driver.

Information concerning the following items is displayed:

- Options installed
- LCM size
- Active ranges
- Number of transfers on each range
- Number of corrected errors on each range
- LCM-resident buffer pool information

To check the current configuration, enter the following at the IRIS system prompt (#):

```
LCMCHECK
```

The system requests that the operator specify the device/file to receive output of the inquiry. The following entries are acceptable:

- <RETURN> - displays output on the operator's terminal.

- IRIS filename - formats output into designated text file. Contents are identical to that displayed on the terminal.

- IRIS device (e.g., $LPT) - sends output to the specified device.

  When output is specified to appear on the operator's terminal, a pause occurs after each twenty lines to allow examination of screen contents. To continue to the next twenty lines, press <RETURN>.

The first items output are the parameters set for the entire LCM, similar to the following:

```
LCMCHECK - Current LCM Status - mon dd, yyyy  hh:mm:ss


Flags:-

LCM ACTIVE
Driver initialized
Power fail O.K.
HAMMING enabled
Battery Backup installed
Battery Backup functional

$LCM revision number n
Size of LCM = p blocks
First block unavailable to IRIS = a

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Space allocation:

Power-Fail Battery Backup = b Blocks
IIB's Begin at Block c, Size = d Blocks
IOB's Begin at Block e, Size = f Blocks
DFT's Begin at Block g, Size = h Blocks
DISCSUBs Begin at Block i, Size = j Blocks
Data Channel Map blk k, Size = m Blocks

Range Table Begins at Block x
Last block used in range table = y
Transfer counting ON
Corrected error counting ON
MAXIMUM Number of ranges = z
```

The flag settings are determined by both hardware and software. For example, LCM ACTIVE indicates the program LCMACTIVATE has been executed and Driver initialized indicates the driver $LCM exists. HAMMING and battery backup are hardware features that are available.

Space allocation indicates the first block and total number of blocks used by the feature. The transfer and error counting is set by the software and is used in producing range table entries as described on the following page.

The flag settings are followed by range table entries and related information, similar to the following:

No.	Range (octal)	Active	LCM Block	Errors	Transfers
0	0/3-3:	YES	0	0	282
1	0/200-200:C	YES	1	0	406
2	0/201-204:U	YES	2	0	114
3	0/301-305:C	YES	6	0	
.					
.					
.					

The following is an explanation of each entry:

- o No. - table entry number (used for reference only)
- o Range (octal) - logical unit number followed by a range of disk block addresses that have been placed on the LCM; to determine the routines represented by the ranges, use DSP to examine the associated disk blocks
- o Active - YES if the specified logical unit is INSTALLed
- o LCM Block - number of the starting block on the LCM that contains the range
- o Errors - number of soft errors that have been encountered trying to read or write data to these LCM blocks
- o Transfers - number of times that information in the specified location was transferred into or out of the LCM

This information can be used as follows:

- o Note the number of transfers on each range. If any range shows a low number of transfers relative to other ranges, use the LCMC program to delete that range. This provides space for other frequently used files or generates a larger LCM buffer pool for dynamic allocation.
- o Check the number of corrected errors on each range. If an LCM block range consistently shows corrected errors, run the LCM diagnostics. If the diagnostics show any errors or if the corrected errors persist, the hardware should be checked.

After the range table has been printed, LCM resident buffer pool information is displayed, similar to the following:

```
TOTAL # OF BLOCKS IN DYNAMIC AREA = 6243
TOTAL # OF HASH BUCKETS (INDEX BLOCKS) = 101
    NUMBER OF HASH BUCKETS CONTAINING  61  ENTRIES = 43
    NUMBER OF HASH BUCKETS CONTAINING  62  ENTRIES = 58
    LCM BLOCK # OF 1ST HASH BUCKET = 1949
```

	-----ENTRIES-----		----WRITE STATISTICS---	
BUCKET	TOTAL	UNUSED	TOTAL	NOT FOUND
1	61	0	1187	136
2	61	0	1224	120
3	61	0	732	141
. . .				

The LCM-resident buffer pool information above gives statistics about how many disk accesses are eliminated by the dynamic allocation. The statistics printed for this dynamic area are primarily for use by POINT 4.

LIBR

LIBR is a system command that produces a listing of the files on the user's assigned or a stipulated logical unit (LU). The listing may be displayed on the user's terminal, sent to a printer, or saved in a file. Information relating to each file is displayed in columns under headings as described below:

LIBR Column Heading	Description
*	File type or language - one letter is displayed in this column to identify the file type. See Table 1-1 for a complete list of file types.
FILENAME[VOL]	Complete filename - if the filename includes a <CTRL-E>, the <CTRL-E> is represented by a colon (:); the portion of the filename following the <CTRL-E> is not displayed.
PROT	File's current protection status - see Section 1.4.2 for an explanation of these two digits.
COST	Amount charged 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.
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.

If the listing is being displayed on the terminal, it may be halted by pressing <CTRL-S>; to resume the display, press <CTRL-Q>. The display may be terminated at any time by pressing <ESC>.

Using LIBR

To start a listing of the user's assigned LU, at the IRIS 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) for which listing is requested

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 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.
	NOTE If the @ parameter follows a filename, it must be preceded by a space.
@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.
*t	Lists only files of the type specified by the file type letter code t (see Table 1-1 for a listing of file types). 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.
- (hyphen)	Lists only master volume of polyfile; does not include volume number.

^ (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 listing. Therefore, any program in the port's active file is lost when an alphabetized listing is requested. The port's active file is not disturbed for nonalphabetized LIBR 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) or a text file (use an exclamation mark (!) to replace 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 a listing of accessible files that meet the following criteria:

- o File resides on logical unit 2 (2/)
- o Filename begins with the letters "TRE"
- o List is to be alphabetized (^)
- o File is a BASIC program (*B), a text file (*T), or a formatted data file (*F)
- o File belongs to any user in account group six (@6)
- o File was last accessed more than 50 hours and less than 80 hours ago (>50<80)
- o List is to be printed on $LPT

Extended LIBR Information

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

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 Figure 2-3). The other two digits are the file type from which the first column of the library listing is derived. See Table 1-1 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.

Disk Block Usage Command

A level 2 or 3 user may obtain a summary of the disk block usage by privilege level on a specified LU. The command form is:

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

or

```
LIBR lu/?
```

where

lu - number of a logical unit

For example, the command

```
#LIBR 0/?
```

produces a display similar to the following:

```
LOGICAL UNIT #0

  DISK USAGE

PRIV        BLOCKS
 3           1175
 2              0
 1              0
 0              0
```

MAGTAPE

MAGTAPE is a utility program that transfers files from disk-to-tape or tape-to-disk. It is an efficient method for making backup copies of files on tape. It can also print a report that lists the processed files.

Files are selected on the basis of parameters entered by the user. The names of the selected files are displayed on pages containing up to 36 filenames for review by the user. Only files listed on pages that have been reviewed by the user will be processed.

Two help modules may be invoked by entering a ? character. The first Help module guides the user in the selection of file parameters for both disk-to-tape and tape-to-disk procedures. The second Help module contains a description of screen modification commands; it applies only to the disk-to-tape procedure.

Separate procedures for MAGTAPE disk-to-tape and tape-to-disk operation are provided in the following subsections.

MAGTAPE Disk-to-Tape Procedure

1. Insert a scratch (blank) tape that is write-enabled with a write-enable ring into the tape unit.

2. If MAGTAPE is resident on LU 0, the default LU, or the user's assigned LU, enter the following at the IRIS system prompt (#):

   ```
   MAGTAPE
   ```

   If the program resides on another logical unit, enter

   ```
   lu/MAGTAPE
   ```

   where
   lu - number of the LU on which the program is resident

   The program requests the device type as follows:

   ```
   INPUT DEVICE:
   ```

3. Enter D to invoke the disk-to-tape module. A parameter selection screen similar to the following is displayed:

```
PORT NUMBER:nn     MAGNETIC TAPE INTERFACE     MAGTAPE n.n  mm/dd/yy

                        Disk ---> Tape

                      INPUT DEVICE: D

                      REPORT OUTPUT: __

                    FILE(S) PREFIX: _____________

                   TYPE OF FILE(S): _

                 SOURCE LOGICAL UNIT: __

                 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, program version, and release date.

   File selection prompts are in the center of the display.

   The three lines at the bottom of the screen are for system/user interaction. Prompts are displayed on the COMMENT line; user input is entered on the COMMAND line; error messages are displayed on the MESSAGE line.

4. Enter the file selection parameters as described below:

   - o REPORT OUTPUT - To print a report containing the names of files transferred to tape, enter a number (1-9) to designate the appropriate line printer, or press <RETURN> to default to $LPT. The report will print at the completion of the MAGTAPE program. To suppress the report, enter N.

   - o FILE(S) PREFIX - Specify the beginning character(s) of the filenames. Entry of <RETURN> selects all files.

   - o TYPE OF FILE(S) - Enter the IRIS file-type code (refer to Table 1-1 for file-type codes).

   - o SOURCE LOGICAL UNIT - Specify the number of the logical unit on which the files reside. Entry of <RETURN> defaults to the user's assigned logical unit.

   - o OPTIONAL PARAMETERS - Enter any valid LIBR command or press <RETURN> to bypass this parameter.

   A Help module may be accessed by entering ? in the first position of any field. Pressing <ESC> in the first field aborts the program; in any other field, <ESC> aborts the current entry and returns the cursor to the previous field.

5. The program asks whether all entries are correct. If not, enter N to return to the selection display. If correct, enter Y to begin the file selection process. A message, WRITING SELECTED FILES, DO NOT DISTURB, is displayed while a listing of the selected files is produced by LIBR.

   When initialization is completed, the program displays the first page of selected filenames. Each page contains up to 36 filenames.

6. Remove the names of any files that are not to be included using the following commands:

Command	Function
A	All - Erases all file names on the current page. None of the files erased will be processed.
E	Execute - Processes the selected filenames (those that remained on each page in the last pass through the workfile); each file is transferred in turn.
P	Page - Stores the filenames remaining on the page (screen), and displays the next group of filenames (unless the end of file has been encountered).

R Restart - Redisplays the current page as it was before any erasure(s).

W Wrap - Stores all remaining filenames in the workfile, then restarts the selection process by renumbering and displaying the first 36 filenames. Because the review process is restarted, every page must again be reviewed. Files listed on a page that is not reviewed will not be processed.

nn (number associated with a filename) - Erases the filename from the page; that file will not be processed. Several numbers separated by commas or spaces, or a range of numbers connected by a hyphen, may be entered at one time. For example,

```
1,4,7-10 20 25
```

causes files numbered 1, 4, 7, 8, 9, 10, 20 and 25 to be erased from the current page.

? Help - Displays a Help module.

7. Review each page. Files listed on pages that are not reviewed are not included in the list to be transferred. To abort the process, press <ESC>. To view a help screen showing the selection commands, enter ?.

8. After all pages have been reviewed and all files that are not to be transferred have been removed, enter the E (Execute) command. The program asks

```
IS THE TAPE MOUNTED (WITH WRITE RING), AND IS DRIVE TURNED ON (Y/N)?
```

9. If the tape has been mounted, enter Y. Transfer begins automatically.

   If the tape has not been mounted, enter N. The following is displayed:

```
LOAD A SCRATCH TAPE, AND THEN TAP 'RETURN'.
```

   Insert a scratch (blank) tape that is write-enabled with a write-enable ring into the tape unit. Transfer begins automatically.

   If a tape cannot be inserted at this time, press <ESC> to abort the program.

10. If the system is unable to access the tape, the following is displayed:

    UNABLE TO ACCESS TAPE DRIVE, CHECK STATUS AND TRY AGAIN!

    To continue, check the tape driver number and make sure the tape is inserted, or abort the program by pressing <ESC>.

11. After the selected files have been processed, the program displays the initial device selection prompt. The user can transfer other files or press <ESC> to return to the IRIS system prompt (#).

MAGTAPE Tape-to-Disk Procedure

The following procedure may be used to restore files from a tape that has been generated by the MAGTAPE program:

1. Insert the MAGTAPE-generated tape (without the write-enable ring) into the tape unit.

    NOTE

    Only tapes generated by the MAGTAPE disk-to-tape process may be used to restore files to disk.

2. If MAGTAPE is resident on LU 0, the applications default LU, or the user's assigned LU, enter the following at the IRIS system prompt (#):

    MAGTAPE

    If the program resides on another logical unit, enter

    lu/MAGTAPE

    where
    lu - number of the LU on which the program is resident

    The program requests the device type as follows:

    INPUT DEVICE:

(procedure continues on next page)

3. Enter T to invoke the tape-to-disk module. A parameter selection screen similar to the following is displayed:

```
PORT NUMBER:nn     MAGNETIC TAPE INTERFACE     MAGTAPE n.n  mm/dd/yy

                        Disk <--- Tape


                      INPUT DEVICE: T

                      REPORT OUTPUT: ___

                   OUTPUT LOGICAL UNIT: _____

                    OVERWRITE MODE ? : ___



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 version, and release date.

File selection prompts are in the center of the display.

The three lines at the bottom of the screen are for system/user interaction. Prompts are displayed on the COMMENT line; user input is entered on the COMMAND line; error messages are displayed on the MESSAGE line.

4. Enter the selection parameters as described below:

   - REPORT OUTPUT - To print a report containing the names of files transferred from tape, enter a number (1-9) to designate the appropriate line printer, or press <RETURN> to default to $LPT. The report will print at the completion of the MAGTAPE program. To suppress the report, enter N.

   - OUTPUT LOGICAL UNIT - Enter the number of the destination logical unit.

   - OVERWRITE MODE - To overwrite any existing files on the output logical unit, enter Y. If existing files on the logical unit are to be maintained, enter N. If the program encounters a file with the same name, the program displays an appropriate message, skips that file, and continues with the next file on the tape. If not enough blocks are available on the destination logical unit, the program halts.

A Help module may be accessed by entering ? in the first position of any field. Pressing <ESC> in the first field aborts the program; in any other field, <ESC> aborts the current entry and returns the cursor to the previous field.

5. When the parameter entries are completed, the program displays

   IS TAPE MOUNTED, AND IS DRIVE TURNED ON LINE (Y/N)?

   If the tape has been mounted, enter Y. Transfer begins automatically.

   If the tape has not been mounted, enter N. The following is displayed:

   LOAD A SCRATCH TAPE, AND THEN TAP 'RETURN'.

   Insert a scratch (blank) tape that is write-enabled with a write-enable ring into the tape unit. Transfer begins automatically.

   If a tape cannot be inserted at this time, press <ESC> to abort the program.

6. If the system is unable to access the tape, the following is displayed:

   UNABLE TO ACCESS TAPE DRIVE, CHECK STATUS AND TRY AGAIN!

   To continue, check that the drive is on-line and the tape is inserted, or press <ESC> to abort the program.

7. After the selected files have been processed, the program displays the initial device selection prompt. The user can transfer other files or press <ESC> to return to the IRIS system prompt (#).

MAIL

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

To mail a message to another port, at the IRIS system 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 1:

```
MAIL 1 -- J. MANAGER, PLEASE INSTALL UNIT 3 <CTRL-G>
```

The following message is received and displayed on port 1, 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 IRIS system prompt. The message prints even if the destination port is not logged on, provided the terminal 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>.

MAKEBIN

MAKEBIN is a BASIC program that converts files from hexadecimal to binary format. Hex files are processed one block at a time. Progress messages indicate which block is in process. MAKEBIN can be run only from the manager or utility accounts.

At the IRIS system prompt (#), enter

```
MAKEBIN
```

If the program is not on LU 0 or the default logical unit, include the logical unit number in the command as follows:

```
lu/MAKEBIN
```

The following example shows the messages displayed and user input required (user input is underlined):

```
PROGRAM TO CONVERT HEX FILE TO BINARY
Hex file to process: filename1
File for binary output (must have "LU/")?: lu/filename2


Processing block 0

Processing block 1
.
.
.
Processing block n

#
```

If filename1 is not a hex file, the 'Hex file to process:' prompt is repeated.

Approximately 30 seconds are required to process each block in the file. Upon completion, the program exits and the IRIS system prompt (#) is displayed.

MAKEHEX

MAKEHEX is a BASIC program that converts files from binary to hexadecimal format. Binary files are processed one block at a time. Progress messages indicate which block is in process. MAKEHEX can be run only from the manager or utility accounts.

At the IRIS system prompt (#), enter

```
MAKEHEX
```

If the program is not on LU 0 or the default logical unit, include the logical unit number in the command as follows:

```
lu/MAKEHEX
```

The following example shows the messages displayed and user input required (user input is underlined):

```
PROGRAM TO CHANGE A FILE TO TRANSMITTABLE HEX FORM
File to translate to hex (must have "LU/"): lu/filename1
File for hex output: filename2


Reading block 0 .... and processing

Reading block 1 .... and processing
.
.
.
Reading block n .... and processing

#
```

If filename1 is not a binary file, the 'File to translate to hex (must have "LU/"):' prompt is repeated.

Approximately 30 seconds are required to process each block in the file. Upon completion, the program exits and the IRIS system prompt (#) is displayed.

MAPACTIVATE

MAPACTIVATE is a system command that activates mapped memory. MAPACTIVATE is normally executed as part of the system manager log-on procedure after installation of logical units.

Mapped memory is available with MARK 4 and 9 systems; the MARK 9 can also have an LCM. If both are present, each must be activated separately.

To activate the mapped memory, enter the following at the IRIS system prompt (#):

```
MAPACTIVATE
```

When MAPACTIVATE is activated, the following is displayed:

```
MAP IS NOW ACTIVE
```

MAPCHECK

MAPCHECK can be used to examine the current status of the MAP driver. The output includes all information set by MAPACTIVATE plus any statistical information accumulated by the MAP driver. MAPCHECK can be used only from the IRIS manager account.

Information concerning the following items is displayed:

- o Size of map
- o Number of reserved blocks
- o Number of partitions
- o Number of blocks in the dynamic area
- o Number of hash buckets (one hash bucket for each index block)

To check the current configuration, enter the following at the IRIS system prompt (#):

```
MAPCHECK
```

The system requests that the operator specify the device/file to receive output of the inquiry. The following entries are acceptable:

- o <RETURN> - displays output on the operator's terminal.
- o IRIS filename - formats output into designated text file; contents are identical to that displayed on the terminal.
- o IRIS device (e.g., $LPT) - sends output to the specified device.

When output is to be displayed on the operator's terminal, a pause occurs after each twenty lines to allow examination of screen contents. To continue to the next twenty lines, press <RETURN>.

The information displayed for the MAP is similar to the following example:

```
                MAPCHECK - Current MAP Status - MAR 17, 1987  13:28:53


     $SYSMAP revision number 5
     Size of MAP = 2048 blocks
     No. of "reserved" blocks = 0


     No. of partitions = 8
     Size of each partition = 8192 words ( = 32 blocks)


     TOTAL # OF BLOCKS IN DYNAMIC AREA = 1568
     TOTAL # OF HASH BUCKETS (INDEX BLOCKS) = 25
         NUMBER OF HASH BUCKETS CONTAINING 62 ENTRIES = 7
         NUMBER OF HASH BUCKETS CONTAINING 63 ENTRIES = 18
         MAP BLOCK # OF 1ST HASH BUCKET = 480

           -----ENTRIES-----          ----WRITE STATISTICS---
     BUCKET   TOTAL     UNUSED        TOTAL          NOT FOUND
     ------   -----     ------  --------------  --------------

          1      61          0            1187            136
          2      61          0            1224            120
          3      61          0             732            141
          .
          .
          .
```

The size of the map includes 256 blocks of main memory used by IRIS. One of the partitions is included in main memory.

The hash buckets are used by the system to determine where to find the information on the map. The statistics printed for this area are primarily for use by POINT 4.

MONITOR

MONITOR is a POINT 4 diagnostic tool and is not intended for customer use.

PF.LUCONV

PF.LUCONV is used to convert polyfiles to IRIS R9 format. For more information, see the IRIS R9 Release Notes.

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, MARK 2E, MARK 3 or MARK 4 systems (excluding the MARK 4E).

PORT EVICT may be used to evict users from their ports as described in the subsection, Port Evict.

PORT commands do not need a password, except for the PORT EVICT command and those PORT commands that are restricted to or by the IRIS system manager. Refer to the IRIS R9 System Manager Manual for information on setting up passwords and changing restrictions.

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

PORT commandstring

where

commandstring - one of the PORT command forms listed in Table 2-6

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

ILLEGAL COMMAND OR SYNTAX

Table 2-6 shows the proper command format and explains the function of each command.

TABLE 2-6. PORT COMMANDS (1 of 3)

Command Format	Description
PORT ACTIVITY	Displays the number of ports currently in use on the system.
PORT BAUD b	Allows change of baud rate at the terminal on the MARK 4E and MARK 5 through MARK 12 systems (see the following subsection, Baud Rate). Cannot be used on POINT 4 MARK 2, MARK 2E, MARK 3 and MARK 4 systems.
PORT n BAUD b	Changes the baud rate to b on port n. Cannot be used on POINT 4 MARK 2, MARK 2E, MARK 3 and MARK 4 systems. Restricted to the IRIS system manager.

TABLE 2-6. PORT COMMANDS (2 of 3)

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 MIGHTY MUX 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 MIGHTY MUX 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 <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 on port on which command is executed. Requires a password. Restricted to the IRIS 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 IRIS system manager or other users designated by the manager.
PORT ALL MONITOR	Same as PORT n MONITOR, except permits the examination of all ports on the system.

(Table continues on next page.)

TABLE 2-6. PORT COMMANDS (3 of 3)

Command Format	Description
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 R9 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 code 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 XOFF command.
PORT XOFF	Disables the transmission of XON and XOFF codes. This is the default condition.

Baud Rate

The baud rate of a terminal on a MARK 4E, and MARK 5 through MARK 12 system may be changed to any of the following baud rates:

110	600*	4800
150	1200	9600
300	2400	19200*

*600 baud is not available on POINT 4 MIGHTY MUX systems with the 19200 baud option.

To change the baud rate on your 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 terminal baud setting must be set to the baud rate specified in the BAUD command. Press <ESC>; if the baud rate was changed successfully, the IRIS system prompt (#) is displayed.

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.

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

PORT n BAUD b

where
- n - number of port
- b - desired baud rate

Unauthorized use of this command generates the message

ILLEGAL COMMAND OR SYNTAX

NOTE

On MARK 2, MARK 2E, MARK 3, and MARK 4 systems, the baud rate is set by jumpering the Peripherals Interface Board.

If the IRIS 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 IRIS system prompt (#) is displayed at the manager's terminal.

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

Port Evict

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

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

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

where
key - password assigned to PORT EVICT (the default is X)
n - number of the port to be evicted

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

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

where
key - password assigned to PORT EVICT (the default is X)

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.

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 only one key. 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.

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 IRIS system prompt (#), enter

   BASIC programname
   <CTRL-C>

   The IRIS system prompt (#) is redisplayed.

2. Enter

   PROTECT <pp>{programname!}

   where

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

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

Using PROTECT with a Key

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

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

   BASIC programname
   <CTRL-C>

   The IRIS system prompt (#) is redisplayed.

2. Enter

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

   where

   pp - protection level given to the program
   programname - name of the program to be protected
   ! - replaces existing program of same name
   key - customer key given on the Pico-N burn report; if omitted, program is PROTECTed, but no special Pico-N is required

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

Patching PROTECTed Programs

Making patches to a PROTECTed BASIC program is a one-way street. That is, if an error is made during the patch process, such as entering a patch with the wrong line number, that program code will be destroyed and cannot be restored. In an effort to help avoid possible errors, we propose the following procedure.

1. Use the COPY command to make a copy of the BASIC program module to be patched. Copy the program to another or the same logical unit, but give it a new name, using the following format at the IRIS system prompt (#):

   ```
   #COPY lu/program.BU=lu/program
   ```

   NEVER patch this backup program !!

2. Create an IRIS text file consisting of all the patch lines of code for the module. Note: REM lines are optional, but may be included for documentation purposes. Be sure to enter each patch line into the text file in the same order shown on the patch listing. If lines are NOT entered in the same order shown, the new check code may not match.

3. Bring the program module into BASIC, load the patch text file, and SAVE the program under the original name.

   ```
   #BASIC lu/program
   LOAD- lu/patchfile
   EXIT
   #SAVE lu/program!
   ```

   IRIS will respond with:

   ```
   SAVED !! CHECK CODE = nnnn
   ```

   CAUTION

   The "EXIT" command must be used. Do not use <CTRL-C>.

4. Keep the copy of the unpatched program available. Also, keep the patch text file on your system and use it to enter any subsequent patches to the same program.

If the above procedure is followed, your BASIC programs should be patched without error and the check codes will agree.

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:

- o File type (see Table 1-1 for a list of file type codes)
- o Protection level
- o Size in blocks
- o Age
- o Last accessed
- o Cost
- o 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 IRIS manager 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:

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

QUERYing a File's Characteristics

To check the characteristics of a file, at the IRIS system 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 on which the file resides

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 same file is queried from the IRIS manager 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.

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

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

QUERYing the User Account Status

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

```
QUERY @
```

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

```
QUERY @lu/
```

where
lu - number of the logical unit to be queried

The resulting display is similar to the following:

```
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

#
```

QUERYPF

QUERYPF is a BASIC program that generates 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:

- o Individual volume display
- o Global display
- o Informational dump

At the IRIS 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.

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 [0-39]
     or <RETURN> for global display
     or -1 for complete dump
     or ESCape to exit to SCOPE:
```

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.

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 is not installed.
- Account number does not match master volume.

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 [0-39]
    or <RETURN> for global display
    or      -1 for complete dump
    or  ESCape to exit to SCOPE:
```

QUERYPF Errors

QUERYPF uses the file 0/POLYFILERRORS to display error messages. If the file is not found, only error codes are output if an error is encountered. 0/POLYFILERRORS can be created by the program BUILDPFERR, which can be run from the utility account (0,2).

Errors may be generated by two functions used by QUERYPF: CALL $VOLLINK and SEARCH. For a list of error codes, see Appendix D.

R7TOR8ACTCONV

R7TOR8ACTCONV is used to convert user accounts from IRIS R7 format to IRIS R8/R9 format. For more information, see the IRIS R9 Release Notes.

RECEIVE - TRANSMIT

RECEIVE and TRANSMIT are used to transmit text files from one system to another using a tri-tail switch. The MUX ports for both CPUs must be set for the same parity, number of stop bits, character length, and baud rate. The RECEIVE-TRANSMIT functions must be performed in the following sequence:

1. Activate the Receiver by setting the tri-tail switch to the receiving CPU, and at the IRIS system prompt (#), enter

   ```
   RECEIVE
   ```

   The following message is displayed:

   ```
   CPU-TO-CPU TEXT FILE RECEIVER PROGRAM
   FILENAME TO BE RECEIVED INTO:
   ```

2. Enter the name of the file to receive the transmitted file. It must be a legal IRIS filename; it need not be the same name as the file to be transmitted. Add an exclamation point (!) to the filename if an existing file on the receiving logical unit is to be overlaid.

3. Activate the Transmitter by setting the tri-tail switch to the sending CPU, and at the IRIS system prompt (#), enter

   ```
   TRANSMIT
   ```

   TRANSMIT asks whether the RECEIVE program is already on the receiving CPU:

   ```
   CPU-TO-CPU TEXT FILE TRANSMITTER
   IS "RECEIVE" PROGRAM ALREADY ON RCVG CPU (Y or N)?
   ```

   If the RECEIVE program is <u>not</u> on the receiving CPU, enter N. The program must be transmitted as explained in the following messages:

   ```
   SWITCH YOUR TERMINAL TO RCVG CPU, LOG ON, ENTER BASIC.
   TYPE CTRL-E CTRL-X
   THEN SWITCH TERMINAL BACK TO XMTG CPU & PRESS RETURN
   ```

   The following is displayed:

   ```
   TO TRANSFER "RECEIVE" TO RCVG CPU, CONNECT XMTG CPU
   TO RCVG CPU YOU WILL HAVE TWO SECONDS AFTER YOU PRESS RETURN...
   ```

   NOTE

   To transmit the RECEIVE program, the transmitting CPU must have the text file version of the program.

   Set the tri-tail switch to the receiving CPU and press <RETURN>. Proceed to step 6.

If the RECEIVE program is already on the receiving CPU, enter Y (the default). The program prompts for the name of the file to be transmitted:

```
TEXT FILE TO BE TRANSMITTED:
```

4. Enter the name of the text file to be transmitted. If the name entered is not a text filename, an error message is displayed and the prompt is repeated.

   If the file is a text file, the following message is displayed:

```
CONNECT XMTG CPU TO RCVG CPU ("RECEIVE" SHOULD BE RUNNING)
IF IT HANGS UP SWITCH TO RCVR, PRESS ESCAPE, RECONNECT XMTG TO RCVR
```

5. Set the tri-tail switch to the terminal position. The file will be displayed as it is transmitted. When the transfer is completed, the following message is displayed:

```
..... END OF TRANSFER .....
```

6. Set the tri-tail switch to the sending CPU and press <ESC> <CTRL-C>.

7. Set the tri-tail switch to the receiving CPU, press <ESC> and do one of the following as appropriate:

   - o If there are no more files to be transmitted, press <CTRL-C>.
   - o If more files are to be transmitted, enter RUN. Repeat the sequence from step 2.

REHASH

REHASH is a system command used to reposition file entries for faster access and to identify deleted entries as "never used". It is run periodically to speed open and close operations and to optimize the INDEX especially where files are created and deleted frequently. REHASH has an option word that can be used to specify the accounts from which REHASH can be run. For more information on the option word, see the IRIS R9 System Manager Manual.

Before doing a REHASH, it is necessary to perform the following steps:

1. Back up the LU.

2. Log onto the manager or other designated account.

3. INSTALL the LU.

4. Run QUERY@ to ensure that there are enough blocks available on the LU. The manager or other designated account must have enough blocks available on the LU to build a temporary file the size of its INDEX. To determine the size of INDEX run QUERY INDEX.

To do a REHASH, enter the following at the IRIS system prompt (#):

```
REHASH
```

The following is displayed:

```
LOGICAL UNIT TO REHASH
```

Enter the number of the desired LU and press <RETURN>. The following is displayed:

```
PLEASE WAIT . . .
```

When the processor completes the REHASH, the following is displayed:

```
ALL DONE
#
```

If a REHASH traps or terminates abnormally, do not use the logical unit. Restore the logical unit from a backup and try REHASH again.

REMOVE

REMOVE is a utility that makes a logical unit unavailable to the operating system. It is generally used when changing disk packs on an independent drive or to take a questionable or damaged logical unit off the operating system. REMOVE has an option word that can be used to specify the accounts from which it can be run. For more information on the option word, see the IRIS R9 System Manager Manual.

If blocks from the logical unit being REMOVEd are allocated to an LCM, REMOVE may take longer to run. Do not halt the processor during the REMOVE process or some data may be lost.

A logical unit may be REMOVEd and then INSTALLed without a SHUTDOWN while an LCM is active.

WARNING!

> If a logical unit is REMOVEd and a different disk partition with the same logical unit number is then INSTALLed on a system where the LCM is active, a loss of data may occur. To avoid this problem, SHUTDOWN and re-IPL before the new disk partition is INSTALLed.

Using REMOVE When Changing Disk Packs

To change a disk pack on an independent drive (i.e., LU 0 resides on another drive), remove all logical units on that drive as follows:

1. Ensure that all files on each logical unit to be REMOVEd are closed.

2. For each LU to be REMOVEd, enter the following at the IRIS system prompt (#):

   ```
   REMOVE lu
   ```

   where
   lu - number of the logical unit

   If the message, LU IN USE is displayed, all users have not logged off. Take appropriate steps to log off any users remaining on the system.

3. Stop the drive.

4. Change the disk pack.

5. Start the drive.

6. INSTALL the logical units.

Using REMOVE to Take Off a Questionable or Damaged LU

Before removing a questionable or damaged logical unit, make sure that the data on that logical unit is backed up and, if possible, salvaged using INSTALL FAST. Once that is done, REMOVE the logical unit as follows:

To REMOVE the logical unit, enter the following at the IRIS system prompt (#):

```
REMOVE lu
```

where

lu - number of the logical unit

RENUMBER

RENUMBER can be used to renumber BASIC programs that are in text file format. It creates a list of text files based on parameters entered by the user. After the list has been created, names can be removed from it, if desired.

The renumbering process numbers the lines in increments of ten; the first line number in the renumbered program is line 10.

If several consecutive lines have the same line number, they will be assigned the same new line number. For example, if all REM statements at the beginning of the text file were given the line number 1, they are all assigned the line number 10 in the renumbered file.

No line number can be less than the previous line number.

To invoke RENUMBER, enter the word RENUMBER at the IRIS system prompt (#). A screen similar to the following is displayed:

```
PORT NUMBER: n    TEXT FILE RENUMBER FACILITY    RENUMBER n.n mm/dd/yy


                  FILES(S) PREFIX: _____________
                  LOGICAL UNIT: __


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

Enter as much of the filename as necessary to identify the desired files. To select all text files on the logical unit, press <RETURN> without entering any prefix.

Enter the logical unit number. <RETURN> defaults to the user's assigned logical unit.

To display the help screen, enter a question mark (?).

The list of files that meet the criteria is displayed. Edit the list, if necessary. To execute the renumber function, enter E.

If a file cannot be renumbered, the message FATAL ERROR is displayed. Press <ESC>. That file is left as it was and processing continues with the next file.

RESTORELU0

RESTORELU0 is used during an IRIS R9 operating system upgrade. For more information, see the IRIS R9 System Configuration Manual.

RETRY

RETRY prints statistics regarding disk, multiplexer, and buffer pool usage.

To invoke RETRY, enter the following at the IRIS system prompt (#):

```
RETRY
```

The following is displayed:

```
This program prints error counts from disk and mux drivers

PRINT WHERE (NAME OR L OR L1 OR CR) ?
```

To display the information on the terminal, press <RETURN>. To save information to a file, enter the filename. To print the information, enter the printer name in the form Ln where L specifies $LPT, L1 specifies $LPT1, etc.

Information similar to the following is displayed:

```
RETRY - RETRY COUNT DISPLAY PROGRAM REV n.n

                    MAR 17, 1987  9:28:27

                    *** RETRY COUNT ***

          ***       DATA        ADDRESS     DATA
          ***      CHECK         CHECK      LATE

          LU/ 0 =    0             0          0
          LU/ 5 =    0             0          0
          LU/ 2 =    0             0          0

          MIGHTY-MUX OVERLOAD ERROR COUNT = 0

Total # Buffer Pool Table Searches                    1112
# Times Buffer Pool Search was unsuccessful            853
Total # DATAPUMP Read requests (incl. XMem)           901
Total # Actual Disk Reads                             901
Total # DATAPUMP Write requests (incl. XMem)          139
Total # Actual Disk Writes                            139
```

The retry counts give the number of times the hardware indicated that an error caused an operation to be retried.

The buffer pool information indicates the efficiency of the buffer pools and is primarily for use by POINT 4.

RUN

A saved IRIS BASIC program may be run (executed) directly without calling up the BASIC interpreter.

RUN may also be used as a BASIC command. For information on its functions, refer to the IRIS R9 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 for the program
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 - number of logical unit

SAVE

The SAVE command is used to store a copy on disk of the BASIC program currently in the port's active file.

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

Using SAVE

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

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

where

```
      pp - optional protection code
$ddd.cc - optional access cost
      lu - number of another LU
       ! - replaces existing program of 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.

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.

SAVE Error Messages

Table 2-7 describes the errors that may occur when using SAVE.

TABLE 2-7. SAVE COMMAND ERROR MESSAGES

Error Message	Description
EMPTY FILE, NOT SAVED	There is no program in the port's active file.
FILENAME IN USE BY DIFFERENT ACCOUNT, NOT SAVED	Program is on another account.
ILLEGAL FILENAME, NOT SAVED	The identifier given is not a valid IRIS filename.
FILENAME IN USE AND NO "!" SUPPLIED, NOT SAVED	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 IRIS 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 unnecessary files or programs.
SYNTAX ERROR IN COST, PROTECTION, OR SIZE, NOT SAVED	A cost greater than $999.99 was specified, or the specified protection code is invalid.

SAVELU0

SAVELU0 is used during an IRIS R9 operating system upgrade. For more information, see the IRIS R9 System Configuration Manual.

SCOPE

SCOPE (System COmmand ProcEssor) is used to invoke all IRIS functions. It displays the IRIS system prompt (#) and executes the program specified following the prompt.

SCOPE is automatically invoked at the end of all IRIS utilities, except those that log off the system, such as SHUTDOWN.

SETTIME

SETTIME is a program that establishes the date and time. It is used if the date and time are not entered when the system is brought up. SETTIME can be used to change the time only from the IRIS manager account.

To set the date and time, enter the following at the IRIS system prompt (#):

```
SETTIME
```

The current settings are displayed in the format

```
MO DA, YEAR  HR: MI: SC
```

Enter each item separately, and press <RETURN> after each input. If there is no change, press <RETURN>.

SETUP

SETUP is used to configure IRIS. For more information, see the IRIS R9 System Configuration Manual.

SHUTDOWN

SHUTDOWN is used to exit the IRIS Operating System in an orderly manner. It forces the contents of any buffers to be written out to the proper location on disk so that no work in progress is lost. It is also used prior to performing backups or running stand-alone programs. SHUTDOWN is normally performed by the IRIS system manager on the master terminal, although the manager can designate access to the command to other accounts and/or ports. For more information on designating access, refer to the IRIS R9 System Manager Manual.

If SHUTDOWN is initiated with extended memory, LCM, or mapped memory activated, it may take longer to complete because updated blocks stored in the additional memory are written out to disk. Do not halt the system during the SHUTDOWN process or some loss of data may occur.

Before a SHUTDOWN command is given, all users must be logged off. This ensures that all accounts are properly updated and that all files are closed.

The command syntax for SHUTDOWN differs depending upon whether a simple SHUTDOWN or a SHUTDOWN to one or more programs is desired.

The command for a simple SHUTDOWN is:

```
SHUTDOWN <CTRL-E>key<CTRL-E>
```

where
key - password for SHUTDOWN (the default is X)

The password does not display on the screen when it is entered. After a simple SHUTDOWN, the account that issued the command is logged off and a message similar to the following is displayed:

```
SYSTEM IS QUIESCENT

# BYE   GROUP n USER n   mmmddd,yyy hh:mm:ss

NET ACCRUED CHARGES          $nn.nn

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

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

To SHUTDOWN to a specific stand-alone program, the format is

SHUTDOWN <CTRL-E>key<CTRL-E>program

To SHUTDOWN to more than one stand-alone program, the format is

SHUTDOWN <CTRL-E>key<CTRL-E>program,program

Most stand-alone programs automatically begin execution at a location specified in the program. To force execution at a certain memory location, the location can be included in the SHUTDOWN command as follows:

SHUTDOWN <CTRL-E>key<CTRL-E>program Xnnnnn

nnnnn - location at which to begin execution

To load DBUG into memory along with the stand-alone program, the format is

SHUTDOWN <CTRL-E>key<CTRL-E>program @nnnnn

nnnnn - location at which to load DBUG

For example, to SHUTDOWN a MARK 2 or 4 to create a bootable streamer tape, the format is

SHUTDOWN <CTRL-E>key<CTRL-E>DISCUTILITY,STBOOTM3{@73000} {X70030}

@73000 - loads DBUG into memory at location 73000 after loading other files

70030 - specifies that execution is to begin automatically at location 70030

When shutting down to a specific program or to more than one program, a message is displayed that is appropriate to the program(s). (For more information on stand-alone programs, see the IRIS R9 System Manager Manual.)

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. For more information on SMbasic, see the SMbasic Reference Manual.

SMRUN

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

SWAPTEST

SWAPTEST is a program that swaps active files in order to test the ability of the disk controller or Lotus Cache Memory to do multi-block transfers reliably.

When a system is installed, SWAPTEST is used in conjunction with EXERCISER to test the complete interaction of the IRIS Operating System, the computer, and the disk. POINT 4 recommends that EXERCISER and SWAPTEST be run overnight and concurrently, each test on several terminals. If no errors or malfunctions occur, these tests will run forever unless aborted by an operator. If an error or a malfunction occurs in a test, the test aborts and the IRIS system prompt (#) is displayed.

To initiate SWAPTEST on one or more terminals, enter the following at the IRIS system prompt (#) on each of the selected terminals:

```
SWAPTEST
```

The following is displayed:

```
#SWAPTEST
.
.
.
```

A vertical row of dots displays slowly on the left side of the screen. The more terminals that are tested, the slower the display of dots.

To end the test, press <CTRL-C>. The following is displayed:

```
USER REQUESTED EXIT
#
```

After the test is completed, run PORT ALL MONITOR to check whether all test ports are still in RUN. If they are at SCOPE (the IRIS system prompt), a hardware malfunction has occurred, and hardware diagnostics should be run.

TRANSMIT

See RECEIVE-TRANSMIT.

U.CHANGE

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

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

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).

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.

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 IRIS system 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 on which U.CHANGE resides

   U.CHANGE displays the following:

```
PORT NUMBER: nn      FILE CHANGE FACILITY      U.CHANGE n.n 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 three lines at the bottom are for system/user interaction. Prompts are displayed on the COMMENT line, user input is entered on the COMMAND line, error messages are displayed on the MESSAGE line.

2. Enter the file selection parameters as described below:

   - o FILE(S) PREFIX - specify the beginning characters of the filenames to be changed. Entry of <RETURN> selects all files except those limited by subsequent parameter entries.

- o TYPE OF FILE(S) - specify the IRIS file type by entering B for BASIC, T for text files, etc. (see Table 1-1 for file types). Press <RETURN> to list all files which meet the other parameters.

- o SOURCE LOGICAL UNIT - specify the number of the LU on which the files to be changed reside. Entry of <RETURN> defaults to the user's assigned LU.

- o 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.

A Help module may be accessed by entering ? in the first position of any field. Pressing <ESC> in the first field aborts the program; in any other field, <ESC> aborts the current entry and returns the cursor to the previous field.

3. The program asks whether all entries are correct. If not, enter N to return to the selection display. If correct, enter Y to begin the file selection process. A message, BUILDING WORK FILE, DO NOT DISTURB, is displayed while a listing of the selected files is produced by LIBR.

   When initialization is completed, the program displays the first page of selected filenames. Each page contains up to 36 filenames.

4. Remove the names of any files that are not to be CHANGEd using the following commands:

Command	Function
A	All - Erases all file names on the current page. None of the files erased will be processed.
E	Execute - Processes the selected filenames (those that remained on each page in the last pass through the workfile) by invoking CHANGE; each file is CHANGEd in turn.
P	Page - Stores the filenames remaining on the page (screen), and displays the next group of filenames (unless the end of file has been encountered).
R	Restart - Redisplays the current page as it was before any erasure(s).
W	Wrap - Stores all remaining filenames in the workfile, then restarts the selection process by renumbering and displaying the first 36 filenames. Because the review process is restarted, every page must again be reviewed. Files listed on a page that is not reviewed will not be processed.

nn (number associated with a filename) - Erases the filename from the page; that file will not be processed. Several numbers separated by commas or spaces, or a range of numbers connected by a hyphen, may be entered at one time. For example

```
1,4,7-10 20 25
```

causes files numbered 1, 4, 7, 8, 9, 10, 20, and 25 to be erased from the current page.

? Help - Displays a Help module.

5. Review each page. Files listed on pages that are not reviewed are not included in the list to be CHANGEd. To abort the process, press <ESC>. To view a help screen showing the file selection commands, enter ?.

6. After all pages have been reviewed and all files that are not to be CHANGEd have been removed, enter the E (Execute) command. The program displays a message as each file is CHANGEd.

7. After the selected files have been CHANGEd, the program asks whether the user wants to CHANGE more files. Entry of Y returns to the parameter selection screen. Entry of N terminates the program and returns to the IRIS system prompt.

U.CONVERT

U.CONVERT converts selected IRIS Business BASIC programs from IRIS R7 format to IRIS R9 format. For more information, see the IRIS R9 Release Notes.

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

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).

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.

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 IRIS system 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 n.n 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 three lines at the bottom are for system/user interaction. Prompts are displayed on the COMMENT line, user input is entered on the COMMAND line, error messages are displayed on the MESSAGE line.

2. Enter the file selection parameters as follows:

   - o FILE(S) PREFIX - specify the beginning characters of filenames to be copied. Entry of <RETURN> selects all files except those limited by subsequent parameter entries.

- o TYPE OF FILE(S) - specify the IRIS file type by entering B for BASIC, T for text files, etc. (see Table 1-1 for file types). Press <RETURN> to list all files that meet the other parameters.

- o SOURCE LOGICAL UNIT - specify the number of the LU on which the files reside. Entry of <RETURN> defaults to the user's assigned LU.

- o 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.

A Help module may be accessed by entering ? in the first position of any field. Pressing <ESC> in the first field aborts the program; in any other field, <ESC> aborts the current entry and returns the cursor to the previous field.

3. The program asks whether all entries are correct. If not, enter N to return to the selection display. If correct, enter Y to begin the file selection process. A message, BUILDING WORK FILE, DO NOT DISTURB, is displayed while a listing of the selected files is produced by LIBR.

   When initialization is completed, the program displays the first page of selected filenames. Each page contains up to 36 filenames.

4. Remove the names of any files that are not to be copied using the following commands:

Command	Function
A	All - Erases all file names on the current page. None of the files erased will be processed.
E	Execute - Processes the selected filenames (those that remained on each page in the last pass through the workfile) by invoking COPY; each file is copied in turn.
P	Page - Stores the filenames remaining on the page (screen), and displays the next group of filenames (unless the end of file has been encountered).
R	Restart - Redisplays the current page as it was before any erasure(s).
W	Wrap - Stores all remaining filenames in the workfile, then restarts the selection process by renumbering and displaying the first 36 filenames. Because the review process is restarted, every page must again be reviewed. Files listed on a page that is not reviewed will not be processed.

nn (number associated with a filename) - Erases the filename from the page; that file will not be processed. Several numbers separated by commas or spaces, or a range of numbers connected by a hyphen, may be entered at one time. For example

```
1,4,7-10 20 25
```

causes files numbered 1, 4, 7, 8, 9, 10, 20, and 25 to be erased from the current page.

? Help - Displays a Help module.

5. Review each page. Files listed on pages that are not reviewed are not included in the list to be copied. To abort the process, press <ESC>. To view a help screen showing the file selection commands, enter ?.

6. After all pages have been reviewed and all files that are not to be copied have been removed, enter the E (Execute) command. The program displays a message as each file is copied.

7. After the selected files have been copied, the program asks whether the user wants to COPY more files. Entry of Y returns to the parameter selection screen. Entry of N terminates the program and returns to the IRIS system prompt.

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.

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).

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.

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 IRIS system 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 on which U.KILL resides

   U.KILL displays the following:

```
PORT NUMBER: nn     FILE DELETION FACILITY      U.KILL n.n 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 three lines at the bottom are for system/user interaction. Prompts are displayed on the COMMENT line, user input is entered on the COMMAND line, error messages are displayed on the MESSAGE line.

2. Enter the file selection parameters as follows:

   - o FILE(S) PREFIX - specify the beginning characters of filenames to be deleted. Entry of <RETURN> selects all files except those limited by subsequent parameter entries.

- o TYPE OF FILE(S) - specify the IRIS file type by entering B for BASIC, T for text files, etc. (see Table 1-1 for file types). Press <RETURN> to list all files that meet the other parameters.

- o LOGICAL UNIT - specify the number of the LU where files are to be deleted. Entry of <RETURN>) defaults to the user's assigned LU.

A Help module may be accessed by entering ? in the first position of any field. Pressing <ESC> in the first field aborts the program; in any other field, <ESC> aborts the current entry and returns the cursor to the previous field.

3. The program asks whether all entries are correct. If not, enter N to return to the selection display. If correct, enter Y to begin the file selection process. A message, BUILDING WORK FILE, DO NOT DISTURB, is displayed while a listing of the selected files is produced by LIBR.

   When initialization is completed, the program displays the first page of selected filenames. Each page contains up to 36 filenames.

4. Remove the names of any files that are not to be KILLed using the following commands:

Command	Function
A	All - Erases all file names on the current page. None of the files erased will be processed.
E	Execute - Processes the selected filenames (those that remained on each page in the last pass through the workfile) by invoking KILL; each file is KILLed in turn.
P	Page - Stores the filenames remaining on the page (screen), and displays the next group of filenames (unless the end of file has been encountered).
R	Restart - Redisplays the current page as it was before any erasure(s).
W	Wrap - Stores all remaining filenames in the workfile, then restarts the selection process by renumbering and displaying the first 36 filenames. Because the review process is restarted, every page must again be reviewed. Files listed on a page that is not reviewed will not be processed.

nn (number associated with a filename) - Erases the filename from the page; that file will not be processed. Several numbers separated by commas or spaces, or a range of numbers connected by a hyphen, may be entered at one time. For example

```
1,4,7-10 20 25
```

causes files numbered 1, 4, 7, 8, 9, 10, 20, and 25 to be erased from the current page.

? Help - Displays a Help module.

5. Review each page. Files listed on pages that are not reviewed are not included in the list to be KILLed. To abort the process, press <ESC>. To view a help screen showing the file selection commands, enter ?.

6. After all pages have been reviewed and all files that are not to be KILLed have been removed, enter the E (Execute) command. The program displays a message as each file is KILLed.

7. After the selected files have been KILLed, the program asks whether the user wants to KILL more files. Entry of Y returns to the parameter selection screen. Entry of N terminates the program and returns to the IRIS system prompt.

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.

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).

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.

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 IRIS system 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 on which U.PROTECT resides

   U.PROTECT displays the following:

```
PORT NUMBER: nn    FILE PROTECT FACILITY    U.PROTECT n.n 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.

   The three lines at the bottom are for system/user interaction. Prompts are displayed on the COMMENT line, user input is entered on the COMMAND line, error messages are displayed on the MESSAGE line.

2. Enter the file selection parameters as follows:

   - o FILE(S) PREFIX - specify the beginning characters of filenames to be protected. Entry of <RETURN> selects all files except those limited by subsequent parameter entries.

   - o LOGICAL UNIT - enter the number of the LU on which files are to be protected. Entry of <RETURN> defaults to the user's assigned LU.

A Help module may be accessed by entering ? in the first position of any field. Pressing <ESC> in the first field aborts the program; in any other field, <ESC> aborts the current entry and returns the cursor to the previous field.

3. The program asks whether all entries are correct. If not, enter N to return to the selection display. If correct, enter Y to begin the file selection process. A message, BUILDING WORK FILE, DO NOT DISTURB, is displayed while a listing of the selected files is produced by LIBR.

   When initialization is completed, the program displays the first page of selected filenames. Each page contains up to 36 filenames.

4. Remove the names of any files that are not to be PROTECTed using the following commands:

Command	Function
A	All - Erases all file names on the current page. None of the files erased will be processed.
E	Execute - Processes the selected filenames (those that remained on each page in the last pass through the workfile) by invoking PROTECT; each file is PROTECTed in turn.
P	Page - Stores the filenames remaining on the page (screen), and displays the next group of filenames (unless the end of file has been encountered).
R	Restart - Redisplays the current page as it was before any erasure(s).
W	Wrap - Stores all remaining filenames in the workfile, then restarts the selection process by renumbering and displaying the first 36 filenames. Because the review process is restarted, every page must again be reviewed. Files listed on a page that is not reviewed will not be processed.
nn	(number associated with a filename) - Erases the filename from the page; that file will not be processed. Several numbers separated by commas or spaces, or a range of numbers connected by a hyphen, may be entered at one time. For example 1,4,7-10 20 25 causes files numbered 1, 4, 7, 8, 9, 10, 20, and 25 to be erased from the current page.
?	Help - Displays a Help module.

5. Review each page. Files listed on pages that are not reviewed are not included in the list to be PROTECTed. To abort the process, press <ESC>. To view a help screen showing the file selection commands, enter ?.

6. After all pages have been reviewed and all files that are not to be PROTECTed have been removed, enter the E (Execute) command.

   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.

7. After the selected files have been PROTECTed, the program asks whether the user wants to PROTECT more files. Entry of Y returns to the parameter selection screen. Entry of N terminates the program and returns to the IRIS system prompt.

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.

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.

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).

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.

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 IRIS system prompt (#), enter

   U.SAVE

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

   lu/U.SAVE

   where
   lu - number of the logical unit on which U.SAVE resides

   U.SAVE displays the following:

```
PORT NUMBER: nn     FILE SAVE FACILITY      U.SAVE n.n  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 three lines at the bottom are for system/user interaction. Prompts are displayed on the COMMENT line, user input is entered on the COMMAND line, error messages are displayed on the MESSAGE line.

2. Enter the file selection parameters as follows:

   - o FILE(S) PREFIX - specify the beginning characters of filenames to be saved.

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

   - o LOGICAL UNIT - specify the number of the LU on which the files reside. Entry of <RETURN>) defaults to the user's assigned LU.

A Help module may be accessed by entering ? in the first position of any field. Pressing <ESC> in the first field aborts the program; in any other field, <ESC> aborts the current entry and returns the cursor to the previous field.

3. The program asks whether all entries are correct. If not, enter N to return to the selection display. If correct, enter Y to begin the file selection process. A message, BUILDING WORK FILE, DO NOT DISTURB, is displayed while a listing of the selected files is produced by LIBR.

   When initialization is completed, the program displays the first page of selected filenames. Each page contains up to 36 filenames.

4. Remove the names of any files that are not to be CHANGEd using the following commands:

Command	Function
A	All - Erases all file names on the current page. None of the files erased will be processed.
E	Execute - Processes the selected filenames (those that remained on each page in the last pass through the workfile) by invoking SAVE; each file is SAVEd in turn.
P	Page - Stores the filenames remaining on the page (screen), and displays the next group of filenames (unless the end of file has been encountered).
R	Restart - Redisplays the current page as it was before any erasure(s).
W	Wrap - Stores all remaining filenames in the workfile, then restarts the selection process by renumbering and displaying the first 36 filenames. Because the review process is restarted, every page must again be reviewed. Files listed on a page that is not reviewed will not be processed.
nn	(number associated with a filename) - Erases the filename from the page; that file will not be processed. Several numbers separated by commas or spaces, or a range of numbers connected by a hyphen, may be entered at one time. For example 1,4,7-10 20 25 causes files numbered 1, 4, 7, 8, 9, 10, 20, and 25 to be erased from the current page.
?	Help - Displays a Help module.

5. Review each page. Files listed on pages that are not reviewed are not included in the list to be SAVEd. To abort the process, press <ESC>. To view a help screen showing the file selection commands, enter ?.

6. After all pages have been reviewed and all files that are not to be SAVEd have been removed, enter the E (Execute) command. The program displays a message as each file is SAVEd.

   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.

   A list of error messages and explanations may be found in the section, SAVE Error Messages (Table 2-7).

7. After the selected files have been SAVEd, the program asks whether the user wants to SAVE more files. Entry of Y returns to the parameter selection screen. Entry of N terminates the program and returns to the IRIS system prompt.

UPGRADE

UPGRADE is used during an IRIS R9 operating system upgrade. For more information, see the IRIS R9 System Configuration Manual.

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:

- o A program has been modified
- o A patch made to a PROTECTed program was done successfully
- o An unauthorized modification has been made
- o A POINT 4-supplied BASIC program is the correct version
- o 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 IRIS system prompt (#), enter

VERIFY programname

To obtain the check code of a program currently in the active file, at the IRIS system 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
```

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:

- o Title page
- o Source code list with line numbers referenced by a GOSUB or GOTO statement is prefixed with a plus sign (+)
- o Symbol table with BASIC variables in alphabetic sequence with associated program line number references; line number may be tagged to indicate usage
- o Channel numbers with line number references followed by BASIC statements with their line number references
- o BASIC keywords with line number references
- o 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 (*)
- o Cross-reference symbols used by XREF

XREF offers a number of different methods of file selection and the option of running the program on a phantom port. The user may:

- o Create a new LIBR work file
- o Enter names of programs to be cross-referenced directly into the work file
- o Recall the work file (EDITSVppp.SAVE) if the procedure was interrupted or other program files need to be cross-referenced

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 (EDITSVppp) and the permanent file (EDITSVppp.SAVE). The temporary file (EDITWKApp) 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 (ppp).

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 review and selection commands.

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

XREF Procedure

1. If XREF resides on LU 0, the default LU, or the user's assigned LU, at the IRIS system 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 on which XREF resides

   A cross-reference parameter selection screen similar to the following 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 name, 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. Prompts are displayed on the COMMENT line, user input is entered on the COMMAND line, error messages are displayed on the MESSAGE line. User input must be entered in upper case.

2. The program asks

   CREATE NEW WORK FILE? (Y/N)

   If an existing work file is to be used, refer to the section, Reusing SAVEd Work File.

   To create a new work file, enter Y.

3. The program prompts

   CREATE 'LIBR' WORK FILE? (Y/N)

   If program filenames are to be entered directly, refer to the section, Direct Filename Entry.

   To create a new LIBR listing, enter Y.

4. Enter the file selection parameters described below:

   - o FILE(S) PREFIX - specify the beginning characters of filenames. Entry of <RETURN> defaults to the user's assigned logical unit.
   - o LOGICAL UNIT - enter the number of the logical unit on which the programs reside. Entry of <RETURN> defaults to the user's assigned LU.
   - o 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

   A Help module may be accessed by entering ? in the first position of any field. Pressing <ESC> in the first field aborts the program; in any other field, <ESC> aborts the current entry and returns the cursor to the previous field.

5. The program asks whether all entries are correct. If not, enter N to return to the selection display. If correct, enter Y.

   XREF prompts for the number of copies to be printed

   ENTER NUMBER OF CROSS-REFERENCE LISTINGS:

6. Enter the desired number of copies of the listings. The program then prompts

   ENTER RECIPIENT'S NAME:

7. Enter the name of the person to receive the listings. A message, FORMATTING SELECTED FILES. DO NOT DISTURB, is displayed while a listing of the selected files is produced by LIBR.

   When initialization is completed, the program displays the first page of selected filenames. Each page contains up to 36 filenames.

8. Remove the names of any files from the screen that are not to be included using the following commands:

Command	Function
A	All - Erases all file names on the current page. None of the files erased will be processed.
E	Execute - Processes the selected filenames (those that remained on each page in the last pass through the workfile) for inclusion in the cross-reference.
P	Page - Stores the filenames remaining on the page (screen), and displays the next group of filenames (unless the end of file has been encountered).
R	Restart - Redisplays the current page as it was before any erasure(s).
W	Wrap - Stores all remaining filenames in the workfile, then restarts the selection process by renumbering and displaying the first 36 filenames. Because the review process is restarted, every page must again be reviewed. Files listed on a page that is not reviewed will not be processed.
nn	(number associated with a filename) - Erases the filename from the page; that file will not be processed. Several numbers separated by commas or spaces, or a range of numbers connected by a hyphen, may be entered at one time. For example 1,4,7-10 20 25 causes files numbered 1, 4, 7, 8, 9, 10, 20, and 25 to be erased from the current page.
?	Help - Displays a Help module.

9. Review each page. Files listed on pages that are not reviewed are not included in the list to be cross-referenced. To abort the process, press <ESC>. To view a help screen showing the file selection commands, enter ?.

10. After all pages have been reviewed and all files that are not to be cross-referenced have been removed, enter the E.

11. The next prompt gives the option to run the cross-reference on a phantom port as described in the section, Running XREF on a Phantom Port.

    After processing is completed, the cross-reference is printed. On completion of printing, the IRIS system prompt is displayed.

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 the section, XREF Procedure.

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 the saved file into a new work file while displaying the message

```
#COPY <00> EDITSVppp!=EDITSVppp.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 the section, XREF Procedure.

After the E (Execute) command is given, XREF sends the listings to the printer and returns to the IRIS system prompt (#).

If the files have already been processed, a screen similar to the following will be displayed:

```
PORT NUMBER: nn   CROSS-REFERENCE SELECT    XREF1 n.n mm/dd/yy


                  --> NO FILES SELECTED


COMMENT:  'XREF' COMPLETE, MORE PROGRAMS TO PROCESS (Y/N) ?
COMMAND:
MESSAGE:
```

To build a new work file, repeat the XREF procedure, starting at Step 2.

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 - number of phantom port

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.

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.

An optional word processing package, the Electronic Office System (EOS) is available under IRIS. This 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 IRIS system 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 IRIS system 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 IRIS system 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 are given below:

- o EDIT commands may be entered using either upper or lowercase letters with the exception of XEND and XKIL, which must be uppercase.
- o EDIT commands must be followed by a <RETURN>.
- o EDIT text strings must be terminated by a string delimiter.
- o A slash (/) is the default text string delimiter.
- o Commands operate only within the current page unless otherwise specified.
- o 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.
- o 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.
- o Commands may be combined into one command string as described in Section 3.1.5.
- o 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, an error message and the EDIT command prompt (*) are displayed as follows:

```
NO DESTINATION FILE!
*
```

Table 3-1 shows the syntax of each EDIT command and provides a description of its functions.

TABLE 3-1. EDIT COMMAND SYNTAX/FUNCTION (1 of 5)

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 (2 of 5)

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).
-nG	Same as nG, except gets the nth page preceding the current page.
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.
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 (3 of 5)

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 4-1. EDIT COMMAND SYNTAX/FUNCTION (4 of 5)

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 5-1. EDIT COMMAND SYNTAX/FUNCTION (5 of 5)

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 IRIS system 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 IRIS system 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, <string1> 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

This appendix contains a glossary of the most commonly used terms under IRIS.

Account - an allocation of system time and disk space provided by the IRIS 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.

Active file - space associated with each interactive port. It is used for swapping and stores a user's current partition between time slices.

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.

Backup - procedure to save the system disk, user programs, data files, etc., to another disk, diskette, or tape.

BASIC - the IRIS Business BASIC language interpreter.

BCD - binary coded decimal.

Bit - one binary digit, which may be a 1 or a 0.

Block - a unit of disk storage; an IRIS block stores 512 8-bit bytes of data.

BTUP - block two utility package. A low-level debugger, which resides in disk block 2 on logical unit 0.

Business BASIC - version of the BASIC language used under IRIS.

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 block zero of each logical unit.

CALLTBL - driver containing tables that link BASIC subroutine names and numbers to discsub numbers.

CONFIG - system file containing information about the current system configuration.

Contiguous file - IRIS data file that requires allocation of sequential blocks on a disk.

CPU - computer's central processing unit.

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 - optional disk-to-disk copy utility for non-POINT 4 disk controllers.

DFT - data file table for each active port.

Dirty page - buffer containing data that has not been written to disk.

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.

DSP - disk service processor, a symbolic machine code editor.

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 LU - logical unit number dedicated to read/write operations for "foreign" disks.

Formatted files - IRIS data file consisting of user-defined fields (items) within each record.

Full configuration - a complete IPL activating the whole system.

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.

IPL - initial program load; brings the IRIS Operating System into memory from disk.

IRIS - Interactive Real-Time Information System.

Kilobyte (KB) - 1024 bytes.

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 - designation used to refer to software on a disk or memory-based partition.

LU - logical unit.

LUFIX - logical unit fixed information table.

LUVAR - logical unit variable information table (specifies the physical disk location).

Megabyte (MB) - depending on usage, approximately one million bytes

Minimum configuration - an IPL procedure which activates the master terminal only.

Mux - multiplexer.

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.

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.

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 - user partition size parameters.

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.

REX - real-time executive, contains system-level modules.

Routine - any sequence of instructions or program statements.

SCO - software change order.

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.

SIR - system initializing routine; performs IPL functions.

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.

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.

Word - 16 bits (2 bytes).

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 may then be released. 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.

In some cases, a typed character may not be accepted by the computer. Such cases are:

- o Input not enabled
- o Transmission error
- o Input buffer full
- o 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.

- o 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.
- o 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-1. TERMINAL COMMAND KEYS (1 OF 3)

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 <ESC> key.
<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.

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 if pressed before <RETURN> is pressed. 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 not 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 BASIC, DSP, or EDIT, a <CTRL-Z> 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 inputting device.
<BREAK>	Same as <CTRL-B>.

In some cases, a typed character may not be accepted by the computer. Such cases are:

- o Input not enabled
- o Transmission error
- o Input buffer full
- o 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.

- o 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.
- o 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

200	NUL	<CTRL-@>	240	BLANK	300	@	340	`
201	SOH	<CTRL-A>	241	!	301	A	341	a
202	STX	<CTRL-B>	242	"	302	B	342	b
203	ETX	<CTRL-C>	243	#	303	C	343	c
204	EOT	<CTRL-D>	244	$	304	D	344	d
205	ENQ	<CTRL-E>	245	%	305	E	345	e
206	ACK	<CTRL-F>	246	&	306	F	346	f
207	BEL	<CTRL-G>	247	'	307	G	347	g
210	BKSP	<CTRL-H>	250	(	310	H	350	h
211	HTAB	<CTRL-I>	251	)	311	I	351	i
212	LF	<CTRL-J>	252	*	312	J	352	j
213	VTAB	<CTRL-K>	253	+	313	K	353	k
214	FF	<CTRL-L>	254	,	314	L	354	l
215	CR	<CTRL-M>	255	-	315	M	355	m
216	SO	<CTRL-N>	256	.	316	N	356	n
217	SI	<CTRL-O>	257	/	317	O	357	o
220	DLE	<CTRL-P>	260	0	320	P	360	p
221	XON	<CTRL-Q>	261	1	321	Q	361	q
222	AUXON	<CTRL-R>	262	2	322	R	362	r
223	XOFF	<CTRL-S>	263	3	323	S	363	s
224	AUXOFF	<CTRL-T>	264	4	324	T	364	t
225	NAK	<CTRL-U>	265	5	325	U	365	u
226	SYN	<CTRL-V>	266	6	326	V	366	v
227	ETB	<CTRL-W>	267	7	327	W	367	w
230	CAN	<CTRL-X>	270	8	330	X	370	x
231	ENDMD	<CTRL-Y>	271	9	331	Y	371	y
232	SUB	<CTRL-Z>	272	:	332	Z	372	z
233	ESC	<CTRL-[>	273	;	333	[	373	{
234	F SEP	<CTRL-\>	274	<	334	\	374	|
235	G SEP	<CTRL-]>	275	=	335	]	375	}
236	R SEP	<CTRL-^>	276	>	336	^	376	~
237	U SEP	<CTRL-_>	277	?	337	_	377	DEL

This table lists ASCII codes with the most significant bit set. Subtract 200 to obtain the 7-bit value.

Appendix C

IRIS COMPONENTS

This appendix provides annotated lists of the components shipped with Revision 9 of the IRIS Operating System and indicates the manual in which each component is documented. Note that when 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).

Table C-1 lists the components on logical unit 0; Table C-2 lists the components on logical unit 5. The manual names are abbreviated in the tables; the abbreviations are defined below:

Abbreviation	Full Title
BASIC	IRIS R9 Business BASIC Manual
LOTUSDU	LOTUS DISCUTILITY Manual
Mgr	IRIS R9 System Manager Manual
M2/4DU	MARK 2/4 DISCUTILITY Document
M3UG	MARK 3 User Guide
Phbk	IRIS R9 Peripherals Handbook
SC	IRIS R9 System Configuration Manual
User	IRIS R9 User Reference Manual

TABLE C-1. LU 0 COMPONENTS (1 of 8)

Name	Manual	Description
ACCOUNTS	-	Account directory - contains account parameters and charges for each user; an ACCOUNTS file is created on logical unit 0 and each assigned logical unit by ACCOUNTUTILITY; includes one supplementary module: USERID
ACCOUNTUTILITY	User	Utility used to maintain user accounts and the ACCOUNTS file; includes one supplementary module: ACTUTIL.1
ALOAD	User	Utility used to merge binary files
ASM	User	Absolute Assembler
BASIC	BASIC	BASIC language editor and lister
BASICTEST	User	BASIC readiness test
BCONVERT	RN	Utility used to convert IRIS R7 programs to the R8/R9 format
BLOCKCOPY	Mgr	Stand-alone program used for block-by-block transfers from one disk to another
BYE	User	Log-on/Log-off utility
CALLTBL	BASIC	System driver containing the translation table used to link subroutine names and numbers used in BASIC CALL statements to the corresponding discsub number
CHANGE	User	Utility used to change file attributes

TABLE C-1. LU 0 COMPONENTS (2 of 8)

Name	Manual	Description
CLEANUP	User	Utility used to realign block usage on a logical unit
CLEANUPX	User	Utility similar to CLEANUP, but in addition, allows movement of data between two unlike logical units
CONFIG	Phbk	System file containing information about the current system configuration
COPY	User	Utility used to copy files
CTR	-	System software diagnostic driver for use by POINT 4
DEC	SC	System driver for decimal arithmetic
DEFS	-	System file containing IRIS software definitions
DISCSUBS	SC	System file containing system subroutines
DMAP	-	Disk map - contains disk block usage for the logical unit; a DMAP file is built for each logical unit by the IPL process, or by INSTALL
DSP	Mgr	Disk Service Processor - machine language editor used to make changes to locations in memory and on disk
DU.LOTUS	LOTUSDU	DISCUTILITY for LOTUS controllers
DU.M3	M3UG	DISCUTILITY for MARK 3 controllers

TABLE C-1. LU 0 COMPONENTS (3 of 8)

Name	Manual	Description
DU.WDI	M2/4DU	DISCUTILITY for systems using Western Digital controllers
EDIT	User	Text file editor
EIS	-	System driver for extended instruction set (alternative for DEC)
FAULTHISTORY	-	Information file for system faults
FAULTPRINT	User	Utility used to print TRAP messages
FLBOOT	Mgr	Utility used to boot software from diskettes; used only by MARK 3 systems
FOREIGN	SC	Driver for reading/writing non-IRIS-generated disks and diskettes
FORMAT	User	Utility used to create formatted or contiguous data files
GUIDE	-	Utility used to give directions for the use of BLOCKCOPY; includes one supplementary module: GUIDE.BLOCKCPY
INDEX	-	File directory for logical unit; contains filename and real disk address (RDA) of each file header; an index is created for each logical unit
INSTALL	User	Utility used to open a logical unit or to create a new one

TABLE C-1. LU 0 COMPONENTS (4 of 8)

Name	Manual	Description
IRIS.INIT	SC	Series of programs used to create the initial configuration of the system; includes the following supplementary modules: IRIS.CONFIG1 IRIS.CONFIG2 IRIS.CONFIG2A IRIS.CONFIG2B IRIS.CONFIG3 IRIS.CONGIG3A IRIS.CONFIG4 IRIS.CONFIG5 IRIS.CONFIG6 IRIS.CONFIG7 IRIS.INIT0 IRIS.INITA IRIS.SMBASIC IRIS.SMHDWR
IRIS.SETTERM	SC	Program used before auto-initialization to specify terminal type
IRIS.START	Mgr	Program invoked by BYE at log-on
IRIS.START0	SC	Program that becomes IRIS.START.IPL after the auto-initialization procedure is finished
IRIS.START.IPL	SC	Program invoked by IRIS at the end of IPL
KILL	User	Utility used to delete files
LCM.LCM	SC	Driver for POINT 4 Lotus Cache Memory (LCM)
LCM.XM	SC	Driver for Extended Memory
LCMACTIVATE	User	Utility used to activate the Lotus Cache Memory or Extended Memory driver

TABLE C-1. LU 0 COMPONENTS (5 of 8)

Name	Manual	Description
LCMC	SC	Utility used to configure the Lotus Cache Memory and Extended Memory drivers; includes three supplementary modules: LCMC.1 LCMC.2 LCMC.3
LCMCHECK	User	Utility used to check the status of the LCM and Extended Memory drivers
LIBR	User	Utility used to list file information for a specified logical unit
LPTN	SC	Driver for printer; used with LPTS
LPTS	SC	System driver for printer
LPTTBL	SC	File containing table of printer characteristics used by LPTN
M2FLBOOT	Mgr	Stand-alone program used to boot software from diskettes; used by POINT 4 MARK 2 and MARK 4 systems
MAIL	User	Utility used to send messages from one port to another
MAPACTIVATE	User	Utility used to activate the mapped memory driver
MAPCHECK	User	Utility used to check the status of the map
MESSAGES	-	File containing standard messages
MK12	SC	Driver for the MARK 12 CPU

TABLE C-1. LU 0 COMPONENTS (6 of 8)

Name	Manual	Description
MMUXM3	SC	Driver for the POINT 4 multiplexer on MARK 2 and MARK 3 systems
MMUXM4	SC	Driver for the POINT 4 multiplexer on MARK 4 systems
MMUXM5	SC	Driver for the POINT 4 multiplexer on MARK 5 through MARK 12 systems
MTA0	Mgr	Driver for magnetic tape units
MTAS	Mgr	System driver for magnetic tape units
MTBOOTM5	Mgr	Stand-alone program used to boot programs from magnetic tape on MARK 5 through MARK 12 systems
PHA	SC	Driver for phantom ports
PORT	User	Utility used to change port attributes and display port activity
PROTECT	User	Utility used to save IRIS BASIC programs in an unlistable format
PZ	-	Page zero software definitions in REX
QUERY	User	Utility used to display file characteristics and account status information
REHASH	User	Utility used to reposition index file entries on a logical unit; identifies the entry slots that have never been used or have been deleted, and permits speedier INDEX access

TABLE C-1. LU 0 COMPONENTS (7 of 8)

Name	Manual	Description
REMOVE	User	Utility used to close a logical unit
REXM3	-	Resident Executive for MARK 2 through MARK 4 systems
REXM5	-	Resident Executive for MARK 5 through MARK 12 systems
RTC	-	Driver for real-time clock
RUN	User	Run-time interpreter used to execute IRIS BASIC programs
RUNMAT	-	Run-time interpreter used to execute IRIS BASIC matrix algebra
SAVE	User	Utility used to save IRIS BASIC programs
SBU.BU	-	SMbasic backup procedures; includes one supplementary module: SBU.GETB
SCOPE	User	System Command Processor - displays the IRIS system prompt (#), which indicates that the system is active and ready for input
SETTIME	User	Utility used to set system date and time
SHUTDOWN	User	Utility used to perform all necessary system shutdown functions such as clearing the buffer pool; provides an orderly transition to a stand-alone operation or shutdown of the system

TABLE C-1. LU 0 COMPONENTS (8 of 8)

Name	Manual	Description
STBOOTM3	Mgr	Stand-alone program used to boot software from streamer tape; for MARK 2 through MARK 4 systems
STBOOTM5	Mgr	Stand-alone program used to boot software from streamer tape; for MARK 5 through MARK 12 systems.
SYMBOLS	-	File containing ASM symbols
SYS.SCHED	-	System driver for the timesharing scheduler
SYSMAP	-	System driver for mapped memory
TERM.name	Phbk	Terminal translation module - each terminal translation module contains translation tables and subroutines for a specific terminal type; the IRIS R9 Peripherals Handbook lists the specific terminals and features
TERMS	-	System driver for terminal translation modules
VERIFY	User	Utility used to generate a check code for IRIS BASIC programs

TABLE C-2. LU 5 COMPONENTS (1 of 5)

Name	Manual	Description
ABASIC	BASIC	Advanced IRIS BASIC language editor; includes the following supplementary modules: ABASIC.1 ABASIC.2 ABASIC.3 ABASIC.4 ABASIC.5 ABASIC.A ABASIC.B ABASIC.XREF
ACCOUNTS	-	Account directory; contains accounting parameters and charges for each user with space on the logical unit
ACS.TDCOPY	User	For POINT 4 use only
ACS.VERIFY	User	A POINT 4 diagnostic tool; not intended for customer use; includes the following supplementary modules: ACS.VERIFY2 ACS.VERIFY3 ACS.VERIFY4
ANALYPF	User	Utility used to analyze polyfile volume structures
BUILDPF	User	Utility used to create and extend polyfiles
BUILDPFERR	User	Utility used to create a message file to be used by ANALYPF, BUILDPF, and QUERYPF
BUILDXF	User	Utility used to create indexed files
CHECKSUM	User	Utility used to generate a checksum for any type of file

TABLE C-2. LU 5 COMPONENTS (2 of 5)

Name	Manual	Description
COREMAP	User	Utility used to display memory allocations
DISPLAY	User	Utility used to display a text file
DMAP	-	Disk map; contains disk block usage for the logical unit
EXERCISER	User	Diagnostic tool
EXTRAPORT	User	Utility used to execute commands on a phantom port
FINDFILE	User	Utility used to search all installed logical units for a specified file
FIXDATE	-	Utility used to change the base system year for IRIS R8 systems; includes one supplementary module: FIXDATEDATA
FORGE	User	Program source code editor for IRIS BASIC; includes the following supplementary modules: FORGE1 FORGE2 FORGE21 FORGE22 FORGE23 FORGE3 FORGE4
GUARD	User	Utility used to allow specified programs access to restricted features
INDEX	-	File directory for the logical unit
KILLPF	User	Utility used to delete polyfiles

TABLE C-2. LU 5 COMPONENTS (3 of 5)

Name	Manual	Description
MAGTAPE	User	Utility used to transfer files between tape and logical units; includes the following supplementary modules: MAGTAPE.LOAD MAGTAPE1 MAGTAPE11 MAGTAPE2 MAGTAPE21
MAKEBIN	User	Utility used to convert MAKEHEXed files back to original format
MAKEBLOCKCOPY	Mgr	Utility used to initialize the stand-alone program BLOCKCOPY
MAKEHEX	User	Utility used to convert file to hexadecimal coded text file, to allow the use of TRANSMIT and RECEIVE
MONITOR	User	Utility program for use by POINT 4 as a diagnostic tool
PF.LUCONV	RN	Utility used to convert polyfiles to IRIS R9 format from format used in previous versions of IRIS
QUERYPF	User	Displays characteristics of polyfiles
R7TOR8ACTCONV	RN	Utility used to convert user accounts from IRIS R7 format to IRIS R8/R9 format
RECEIVE	User	Utility used to receive a text file transmitted from another system via a tri-tail switch
RENUMBER	User	Utility used to renumber IRIS Business BASIC source files; includes the following supplementary modules: RENUMBER1 RENUMBER2 RENUMBER3

TABLE C-2. LU 5 COMPONENTS (4 of 5)

Name	Manual	Description
RESTORELU0	SC	Utility used to restore logical unit 0 information during an upgrade; includes one supplementary module: RESTORELU0A
RETRY	User	Utility used to list the number of unsuccessful disk access attempts for each logical unit, and other runtime statistics
SAVELU0	SC	Utility used to save information on logical unit 0 during an upgrade of IRIS; includes the following supplementary modules: SAVELU0A SAVELU0.revno
SETUP	SC	Utility used to configure IRIS; includes the following supplementary modules: SU.MSG SUnnn.nn where nnn.nn - specific module number
SWAPTEST	User	A diagnostic tool used to detect swapping errors
TRANSMIT	User	Utility used to transmit a text file to another system via a tri-tail switch
U.CHANGE	User	Utility used to change the protection level of selected files; includes one supplementary module: U.CHANGE1
U.CONVERT	RN	Utility used to convert selected BASIC programs from an IRIS R7 format to an IRIS R8/R9 format; includes one supplementary module: U.CONVERT1

TABLE C-2. LU 5 COMPONENTS (5 of 5)

Name	Manual	Description
U.COPY	User	Utility used to copy selected files; includes one supplementary module: U.COPY1
U.KILL	User	Utility used to delete selected files; includes one supplementary module: U.KILL1
U.PROTECT	User	Utility used to protect selected BASIC program files; includes one supplementary module: U.PROTECT1
U.SAVE	User	Utility used to save selected BASIC program files; includes one supplementary module: U.SAVE1
UPGRADE	SC	Utility used to upgrade the utilities logical unit (logical unit 5); includes the following supplementary modules: UPGRADE1 UPG.000 UPG.999 UPG.UTILITIES
XREF	User	Utility used to produce a cross-reference of variables, line numbers, and tokens in an IRIS BASIC program; includes the following modules: XREF1 XREF2 XREF3 XREF4 XREF5 XREF6 XREFA XREFB

Appendix D

POLYFILE ERROR CODES

The polyfile utilities ANALYPF, BUILDPF, and QUERYPF, may report errors based on CALL $VOLLINK or SEARCH error codes. When an error is encountered, the program displays the error code and, if the file POLYFILERRORS exists, appropriate error message. The utility is then terminated.

If an error is encountered by CALL $VOLLINK, a message similar to the following is displayed:

```
Call status = n {message}
```

where
n - error code

Table D-1 lists CALL $VOLLINK error codes and messages.

If an error is encountered by SEARCH, a message similar to the following is displayed:

```
Search status = n {message}
```

where
n - status code

Table D-2 lists SEARCH status codes and messages.

TABLE D-1. CALL $VOLLINK ERROR CODES

CALL $VOLLINK Status	Description
01	Bad channel #
02	File not being built
03	Bad volume #
04	File C1 is not a polyfile
05	Bad filename
06	Bad variable type
07	Bad number
08	Volume preexists on LU; may be part of another polyfile
09	File C0 not found
10	No nodes
11	Volume already defined for this polyfile
12	Volume v not found
13	Account numbers differ
14	Volume in data file table (DFT) but not on disk
15	No available volume numbers
16	Volume not defined; it is missing
17	P does not immediately succeed S in VDT
18	P is not an array
19	P is not dimensioned at least P[10]
20	File C0 is not contiguous
21	File C1 is open elsewhere
22	File C0 is already a polyfile
23	(No longer used)
24	(No longer used)
25	Cannot move volume 0
26	Illegal move operation
27	(No longer used)
28	(No longer used)
29	Volume numbers do not match

TABLE D-2. SEARCH MODE STATUS CODES

SEARCH Mode Status	Description
0	No error, operation successful
1	Operation not successful
2	End of directory
3	End of data
4	File not indexed
5	Polyfile structure error
6	Directory number not in sequence
7	File is not contiguous
8	Volume is already indexed
9	Illegal key length (less than 2 or greater than 121)
10	Too many directories (limit is 64 per polyfile)
11	Volume not found (possible structural error)
12	Volume (built) too small
13	Directory not found
14	File not indexed
15	Data volume number is less than the preexisting data volume
16	Data volume map request not consistent with preexisting volumes
17	Data volume record length does not match that of the polyfile
18	Block record out of range
19	Record was not allocated (already released)
20	Volume has no map

Appendix E

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.

Ask the IRIS 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.

E.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 Section 2) 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.

E.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.

   - o Directory 1 with 100 keys each 10 bytes long
   - o Directory 2 with 100 keys each 31 bytes long
   - o 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 R9 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.

E.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 IRIS system prompt (#), enter

BUILDPF

BUILDPF responds

BUILDPF - Build Polyfile Utility

The response shown below will give the results specified in Section E.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 automatically built first by the BUILDPF program.

```
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

The 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 [ 0-39; <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 <RETURN> is entered in response to the logical unit prompt, BUILDPF returns to the IRIS system prompt and the creation of a polyfile is completed.

E.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.

E.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.

E.2.2 Extending Polyfile Exercise

In this exercise, the responses given for the prompts correspond to the specifications given in Section E.2.1. To extend a polyfile, at the IRIS system 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 [ 0-39; <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 [ 0-39; <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 [ 0-39; <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 [ 0-39; <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 QUERYPF in Section 2. Then, use KILLPF to delete the polyfile.

INDEX

INDEX

COMMENT SHEET

MANUAL TITLE IRIS R9 User Reference Manual

PUBLICATION NO. SM-030-0034 REVISION 05

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 THI WE

POINT DATA CORPORATION

15442 Del Amo Avenue
Tustin, CA 92680
(714) 259-0777