Tektronix®
COMMITTED TO EXCELLENCE

4907

FILE MANAGER

OPERATOR'S MANUAL

Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97077

Manual Part No.
070-2380-01

First Printing MAR 1978
This Printing FEB 1979

WARRANTY

Tektronix warrants to the original purchaser that this product and options, excluding customer supplied equipment, is free from defects in materials and workmanship, under normal use, for a period of one (1) year from the date of shipment. Tektronix will, at its option, repair or replace the product if Tektronix determines it is defective within the warranty period, and it is returned, freight prepaid, to a Tektronix Service Center.

In the forty-eight (48) contiguous United States and the District of Columbia, and in other areas where Tektronix normally has service available for this product, Tektronix will provide on-site warranty service at no charge during the first ninety (90) days from the date of shipment.

Tektronix shall be under no obligation to furnish warranty service if:

a. Attempts to repair or service the product are made by personnel other than Tektronix service representatives.
b. Modifications are made to the hardware or software by personnel other than Tektronix representatives.
c. Damage results from connecting the product to incompatible equipment.

There is no implied warranty of fitness for a particular purpose. Tektronix is not liable for consequential damages.

This instrument, in whole or in part, may be protected by one or more U.S. or foreign patents or patent applications. Information provided on request by Tektronix, Inc., P.O. Box 500, Beaverton, Oregon 97077.

PRODUCT 4907, 4907 Option 30, and 4907 Option 31

This manual supports the following versions of this product: BO10101 and up

MANUAL REVISION STATUS

REV.	DATE	DESCRIPTION
@	3/78	Original Issue
A	11/78	Revision and correction
@	2/79	New pages
A	2/79	Revision
B	2/79	Revision

CONTENTS

Section 1 **GENERAL DESCRIPTION** **Page**

Section 2 **GENERAL OPERATION**

Section 5 (cont)

TABLES

ILLUSTRATIONS

SECTION 1

CONTENTS

Section 1 **GENERAL DESCRIPTION**

Page

Section 1

GENERAL DESCRIPTION

The 4907 File Manager, the 4050 Series flexible disc storage system, adds several new commands to the Graphic System's BASIC vocabulary. These commands, when used with regular Graphic System commands, allow:

- File naming
- File security with passwords
- Automatic increases in file space when necessary
- File copying
- Multiple file access
- Recording time and date of all file activities
- File renaming
- Five file storage levels
- Fast access within files with 'random' access

The 4907 File Manager increases the potential and versatility built into your Graphic System. You'll find this product a valuable and efficient tool in all your information storage activities.

About This Manual

This manual explains how to operate the 4907 File Manager.

In order to use this manual and the 4907 File Manager effectively, you must have a working knowledge of the 4050 Series Graphic System. Most of the descriptions, procedures and terms in this manual have been written on that basis.

Details of data transfer (I/O) and related subjects not covered in this manual are fully described in the 4050 Series Graphic System Reference Manual.

Your Graphic System and 4907 disc drives, together, are referred to as "the system" throughout this manual. Your Graphic System is often called a "host." Although the Graphic System is generally purchased separately, it is considered part of 'the system' for purposes of discussion.

Take time to familiarize yourself with the following information in the manual before attempting to operate the system.

FRONT PANEL CONTROLS AND INDICATORS (in Section 1)

This portion of the GENERAL DESCRIPTION section describes the switches and indicators on the front of your system controller and disc drives.

GENERAL OPERATION (Section 2)

This section outlines the steps required before the system can be used, including GPIB cable & ROM pack installation, power up, and disc loading. It also shows you the steps that are necessary to create and use your first files. Review the GLOSSARY (Appendix D) to understand the terms used.

STORAGE STRUCTURE (Section 3)

This section describes the concepts of file storage. It is necessary to understand these concepts to write file identifiers (F.I.s), which are necessary parts of many commands.

HOW TO WRITE A COMMAND (Section 4)

This section describes the basic rules for writing commands. It shows the general order of the fields and what they do. It also describes the terms used in individual command descriptions.

HOW TO WRITE A FILE IDENTIFIER (in Section 4)

This part of Section 4 describes the rules for writing file identifiers. A file identifier, or F.I., is required in many commands. It helps to deliver information like an address on an envelope, but it can also create the destination (file). In short, the F.I. tells the Graphic System which file is involved in the operation and where it is to be created or where it can be found.

COMMAND DESCRIPTIONS (Section 5)

4907 File Manager commands are described in COMMAND DESCRIPTIONS.

Questions and Answers

The following questions and answers may help you in using the manual.

Q. HOW DO I CREATE A FILE?

A. Look in GENERAL OPERATION (Section 2) for the required steps.

Q. WHAT KIND OF FILE SHOULD I USE?

A. 4907 File Manager files are random or sequential and can contain either ASCII or binary data or programs. See RANDOM AND SEQUENTIAL FILES in GENERAL OPERATION (Section 2). Also see the descriptions of WRITE and PRINT in the 4050 Series Graphic System Reference Manual.

Q. HOW CAN I BE SURE I'M ENTERING COMMANDS IN CORRECT ORDER?

A. Study the GENERAL SEQUENCE FLOW CHART in Section 2. You can see what the command parameters and prerequisites are, if any, by reading the command description.

Q. WHAT IS AN F.I. AND WHAT DOES IT DO?

A. The F.I., or file identifier, is used in many commands to tell the system what the name of the file is and where it is to be placed or where it may be found if it already exists. See STORAGE STRUCTURE (Section 3) and HOW TO WRITE A FILE IDENTIFIER in Section 4.

Q. WHAT IS A CURRENT DEVICE?

A. Commands containing logical file numbers or F.I.'s, but no device addresses, will be directed to the "current device." This device must be specified earlier in a UNIT or CALL "UNIT" command.

Q. WHAT IS THE CURRENT LIBRARY?

A. See CALL "USERLIB" command description in Section 5.

Q. WHAT HAPPENS IF INCORRECT COMMANDS ARE ATTEMPTED?

A. You will generally get an error message if incorrect commands are attempted. If strange results occur even though correct commands have been executed, the cause could include:

- accessing the wrong device
- not executing a UNIT command
- opening a file incorrectly
- using incorrect variables or variables already active in the Graphic System
- not executing an INIT when necessary

In case of problems, check the command sequences required as well as the individual operations. Be sure the problem is not caused by incorrect use of the Graphic System or faulty programming.

Q. HOW DO I GET DATA IN AND OUT OF MY FILES?

A. See SAMPLE I/O in Section 2.

Q. WHAT ABOUT PROGRAMMING?

A. All normal Graphic System programming rules apply. No regular BASIC commands are disabled. See the program examples in INPUT and READ command descriptions in Section 5. Also see SAMPLE PROGRAMS, USING PRINT AND INPUT IN A PROGRAM, USING WRITE AND READ IN A PROGRAM, and SAMPLE I/O PROCEDURES in the appendices.

Q. WHAT ABOUT EXTERNAL DEVICES SUCH AS PRINTERS?

A. The system does not provide for "spooling;" that is, the devices or disc controller cannot listen or talk directly to other external devices. All interaction between the system and peripheral devices must go through the Graphic System.

Q. HOW LARGE SHOULD I MAKE MY FILES OR FILE RECORDS?

A. See the CREATE command description in Section 5.

Q. HOW CAN I CHECK TO SEE HOW MUCH SPACE IS AVAILABLE ON A DISC?

A. See CALL "DSTAT" command description.

Q. HOW DO I RECOVER OPERATION IF EXECUTION IS HALTED AND AN ERROR MESSAGE APPEARS?

A. Most error messages describe the error involved. For more details, see ERROR MESSAGES AND RECOVERY PROCEDURES (Appendix A).

Q. CAN I USE THE 4907 WITH ANY GPIB TALKER?

A. No. The GPIB device must be a controller; that is, talker and listener.

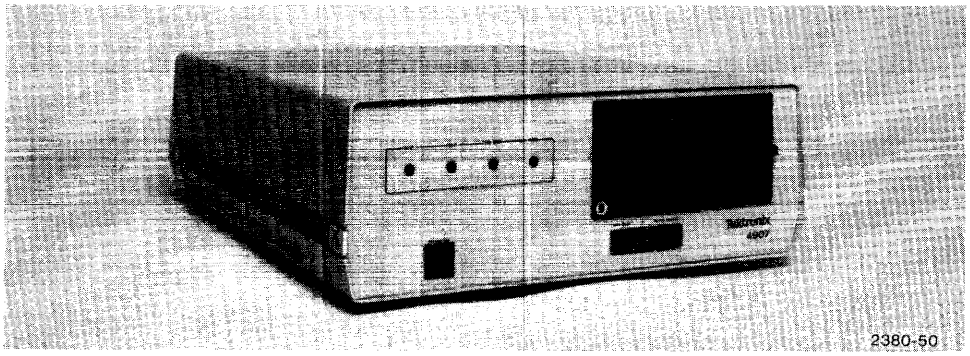


Figure 1-1. 4907 (Single Drive).

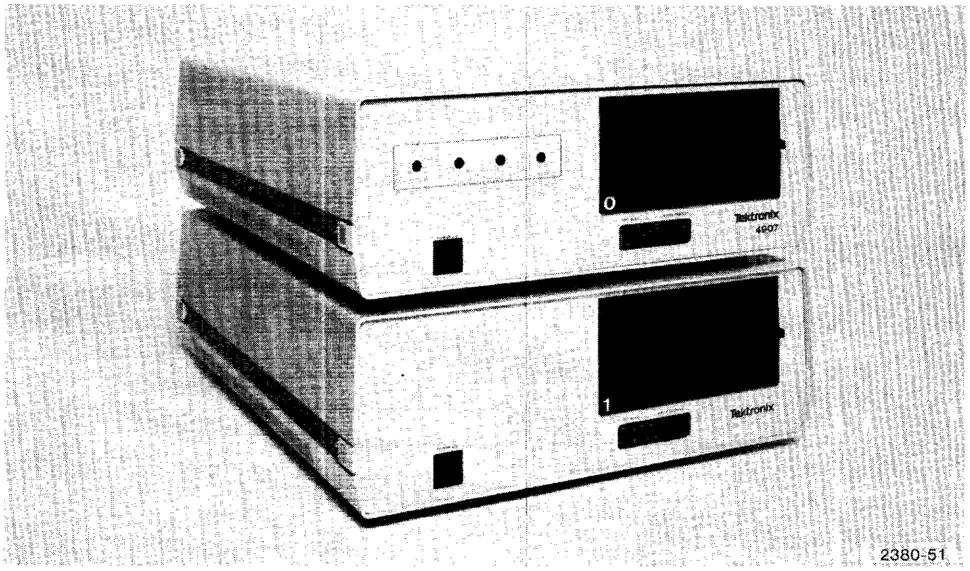


Figure 1-2. 4907 Option 30 (Two Drives).

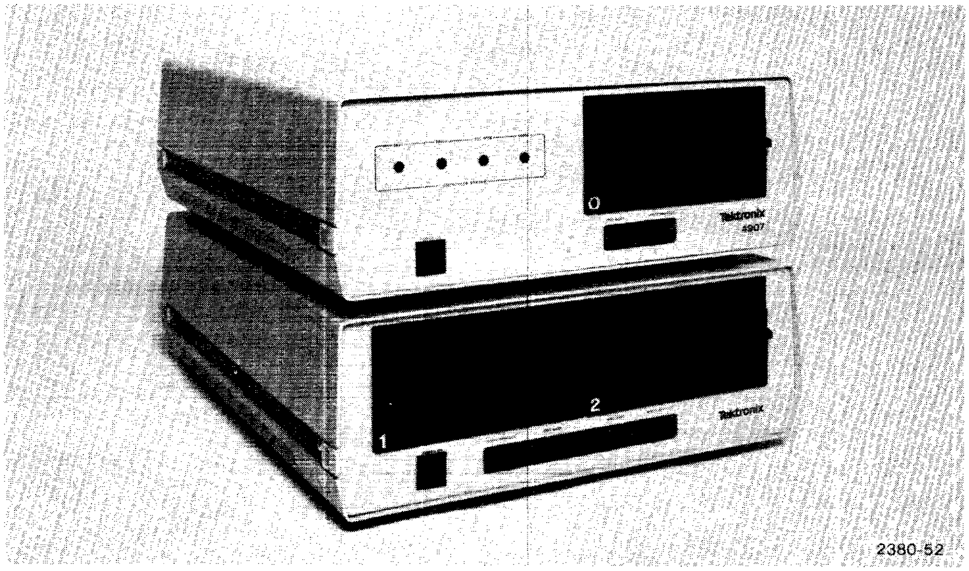


Figure 1-3. 4907 Option 31 (Three Drives).

4907 CONFIGURATION

All 4907 File Managers are designed for use with a TEKTRONIX 4050 Series Graphic System, which controls all storage activity (except for those procedures controlled by the front panel switches on the disc drives and disc controller).

The 4907 File Manager includes a File Manager ROM (Read-Only Memory) pack and from one to three flexible disc drives.

The ROM pack, which plugs into the Graphic System, provides extra BASIC commands for use by the Graphic System. These commands are transmitted from the Graphic System keyboard or program through a GPIB cable to the disc controller in the controller cabinet. The controller activates the single flexible disc drive in that cabinet and, if part of the system, one or two more disc drives in a second cabinet. These drives contain the flexible magnetic discs used for data storage.

The 4907 File Manager is shipped in one of three configurations:

- 4907 (single disc drive).
- 4907 Option 30 (two disc drives).
- 4907 Option 31 (three disc drives).

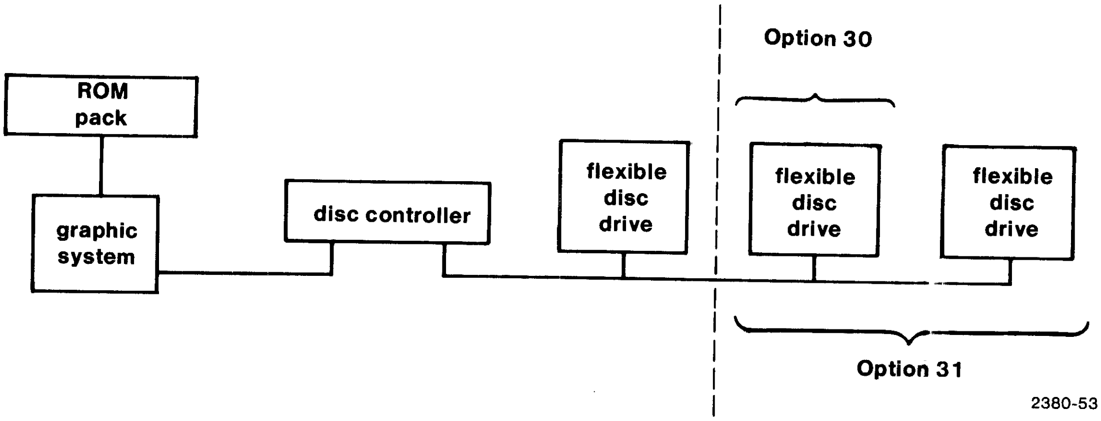


Figure 1-4. 4907 Configuration.

FRONT PANEL CONTROLS AND INDICATORS

The following paragraphs illustrate the functions of the front panel controls and indicators for all configurations of the 4907 File Manager.

4907 Controller and Single Flexible Disc Drive Cabinet

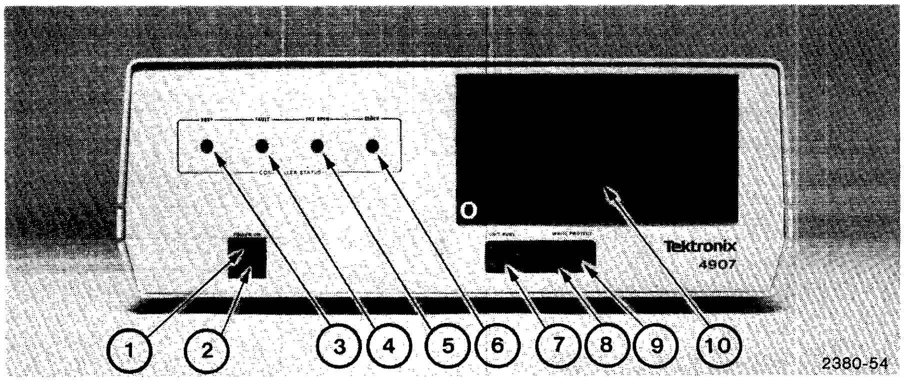


Figure 1-5. Front Panel Controls and Indicators for 4907 Main Cabinet

1. POWER INDICATOR.

2. POWER SWITCH.

3. BUSY INDICATOR. If this light is on, the system is performing a disc operation.

4. FAULT INDICATOR. If this light is on, the system is inoperative. If restart efforts fail, see the 4907 File Manager Service Manual or contact Tektronix service personnel.

5. FILE OPEN INDICATOR. If this light is on, one or more files on a disc are open. Some commands cannot be executed if files are open. See Prerequisites in COMMAND DESCRIPTIONS (Section 5).

6. CLOCK INDICATOR. If this light is on, the system clock must be set.

7. BUSY INDICATOR. If this light is on, this disc drive is in use.

8. WRITE PROTECT SWITCH. A disc may also be write-protected by removing the tape from write-protect hole on the disc (see Figure 2-4 in Section 2, General Operation).

9. WRITE PROTECT INDICATOR. If this light is on, the device or flexible disc is in the write-protect state.

10. DRIVE DOOR RELEASE. Push in to open the drive door.

4907 Option 30 Single Flexible Disc Drive Cabinet

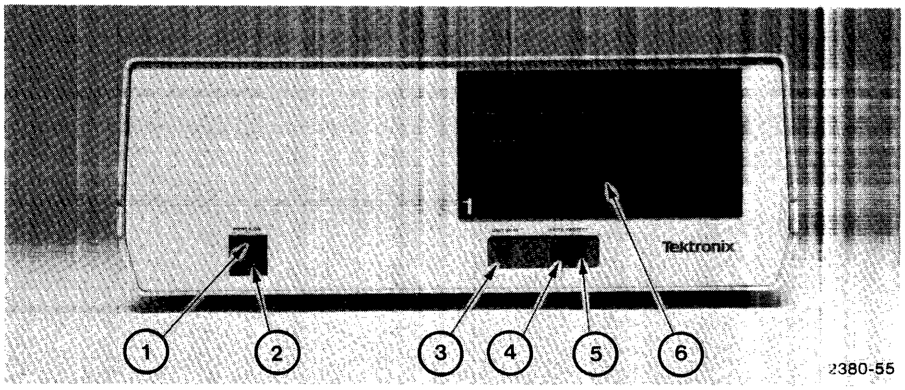


Figure 1-6. Front Panel Controls and Indicators for 4907 Option 30 Single Drive Cabinet.

1. POWER INDICATOR. Even though indicator may be lit, the unit becomes inoperative if power switch on the main cabinet is turned off.

2. POWER SWITCH.

3. BUSY INDICATOR. If this light is on, this disc drive is in use.

4. WRITE-PROTECT SWITCH. Disc may also be write-protected by removing the tape from the write-protect hole on the disc.

5. WRITE-PROTECT INDICATOR. If this light is on, the device or flexible disc is in the write-protect state.

6. DRIVE DOOR RELEASE. Push in to open the drive door.

4907 Option 31 Dual Flexible Disc Drive Cabinet

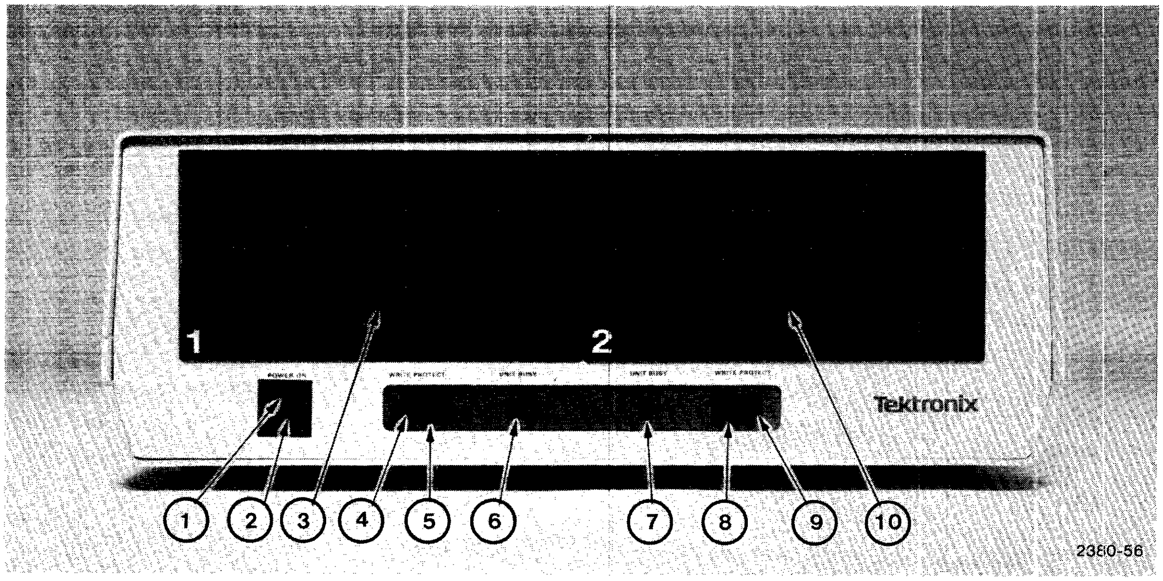


Figure 1-7. Front Panel Controls and Indicators for 4907 Option 31 Dual Drive Cabinet.

1. POWER INDICATOR.

2. POWER SWITCH for both drives.

3. Left DRIVE DOOR RELEASE. Push in to open the drive door.

4. WRITE-PROTECT INDICATOR for left drive. If this light is on, device or flexible disc is in the write-protect state.

5. WRITE-PROTECT SWITCH for left drive.

6. BUSY INDICATOR. If this light is on, the system is performing an operation on the left drive.

7. BUSY INDICATOR. If this light is on, the system is performing an operation on the right drive.

8. WRITE-PROTECT SWITCH for right drive.

9. WRITE-PROTECT INDICATOR for right drive. If this light is on, device or flexible disc is in the write-protect state.

10. Right DRIVE DOOR RELEASE. Push in to open the drive door.

STANDARD/OPTIONAL ACCESSORIES

All 4907 File Manager systems come with the standard accessories listed directly below. The systems with Option 30 or Option 31 contain additional standard accessories which are listed under those headings.

4907 Standard Accessories

4907 File Manager Operator's Manual
Power Cord
1 Flexible Disc
Box of Cleaning Pads (10)
GPIB Cable (2 meters long)
4907 Installation Guide
4907 File Manager Pocket Reference Card
4907 File Manager ROM Pack

4907 Optional Accessories (All Options)

Box of Flexible Discs (10)
4907 Service Manual
GPIB Cable (4 meters long)
Flexible Disc Drive Service Manual
Alignment Disc

4907 Option 30 Standard Accessories

Power Cord
Interconnect Cable
Strain Relief Bracket
Clamp
1 Flexible Disc
Box of Cleaning Pads (10)

4907 Option 31 Additional Standard Accessories

Power Cord
Interconnect Cable
Strain Relief Bracket
Clamp
2 Flexible Discs
Box of Cleaning Pads (10)

4907F32 Field Upgrade Kit Standard Accessories

This kit is used to add a drive to a two-drive system; this creates a system identical to the 4907 Option 31. The 4907F32 kit includes these accessories:

1 Flexible Disc
Box of Cleaning Pads (10)

The 4907F32 Kit includes these components:

Drive Kit
Front Panel
Wire Kit
Ribbon Cable, 50 Conductor
Ribbon Cable, 40 Conductor
Write-Protect Switch with Bezel
LED for Write-Protect Switch
LED (Busy) with Recessed Washer
Cable Ties (2)

SECTION 2

CONTENTS

Section 2 **GENERAL OPERATION** Page

Section 2

GENERAL OPERATION

INTRODUCTION

This section outlines the steps you must perform before the 4907 File Manager can be used, including GPIB cable and ROM pack installation, power up, and disc loading. It also shows the steps that are necessary to create and use your first files. Review the GLOSSARY (Appendix D) to understand the terms used.

PREPARATION

Certain preparatory steps must be performed before the 4907 File Manager can be used. First, if necessary, you must change the address strap and line voltage. Next, install the line cord or cords, ribbon cable (not applicable to the single disc drive 4907), ROM pack, and GPIB cable. Last, power up the system, load the discs and execute performance checks.

When the procedures described in the 4907 Installation Guide have been performed, the File Manager is ready for use.

NOTE

Because GPIB cabling, ROM pack installation, power up, and disc loading are regularly repeated operations, they are included in this manual as well as in the 4907 Installation Guide.

GPIB Cable and ROM Pack Installation

1. Turn off the Graphic System.

2. Connect the GPIB cable from the rear of the Graphic System to the rear of the 4907 main cabinet and tighten both screws on the connector. (Figure 2-1).

3. Plug the 4907 File Manager ROM pack into any available slot in the Graphic System back pack (Figure 2-2) or into a ROM Expander. BE SURE THE GRAPHIC SYSTEM IS TURNED OFF.

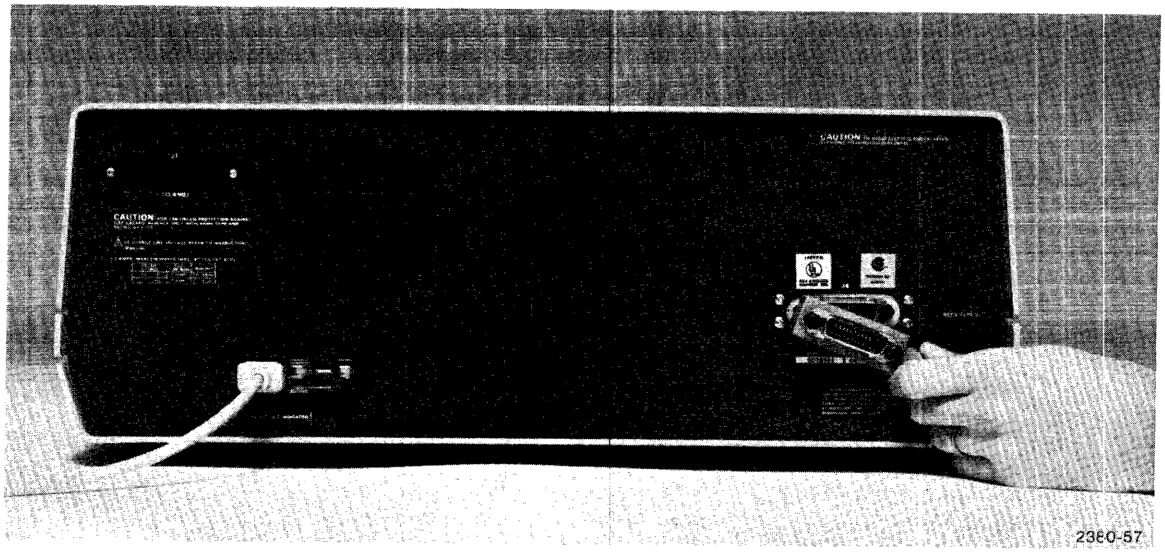


Figure 2-1. Connecting GPIB Cable to Rear of 4907 Main Cabinet.

CAUTION

Inserting any device into or removing any device from a Graphic System backpack slot or a ROM Expander slot when power is on may cause memory to be erased.

CAUTION

If an OPTION 10 (RS-232 PRINTER INTERFACE) ROM pack is in any ROM slot, the File Manager ROM pack must be inserted in the leftmost slot (as you sit at the Graphic System Keyboard, facing the display).

NOTE

The ROM pack may also be plugged into any available slot in any 4050 series ROM Expander, but only one 4907 File Manager ROM pack may be plugged into either a ROM expander or a backpack on the same Graphic System. Installing more than one of these ROM packs sets up a conflict since each 4907 would be considered GPIB device 0.

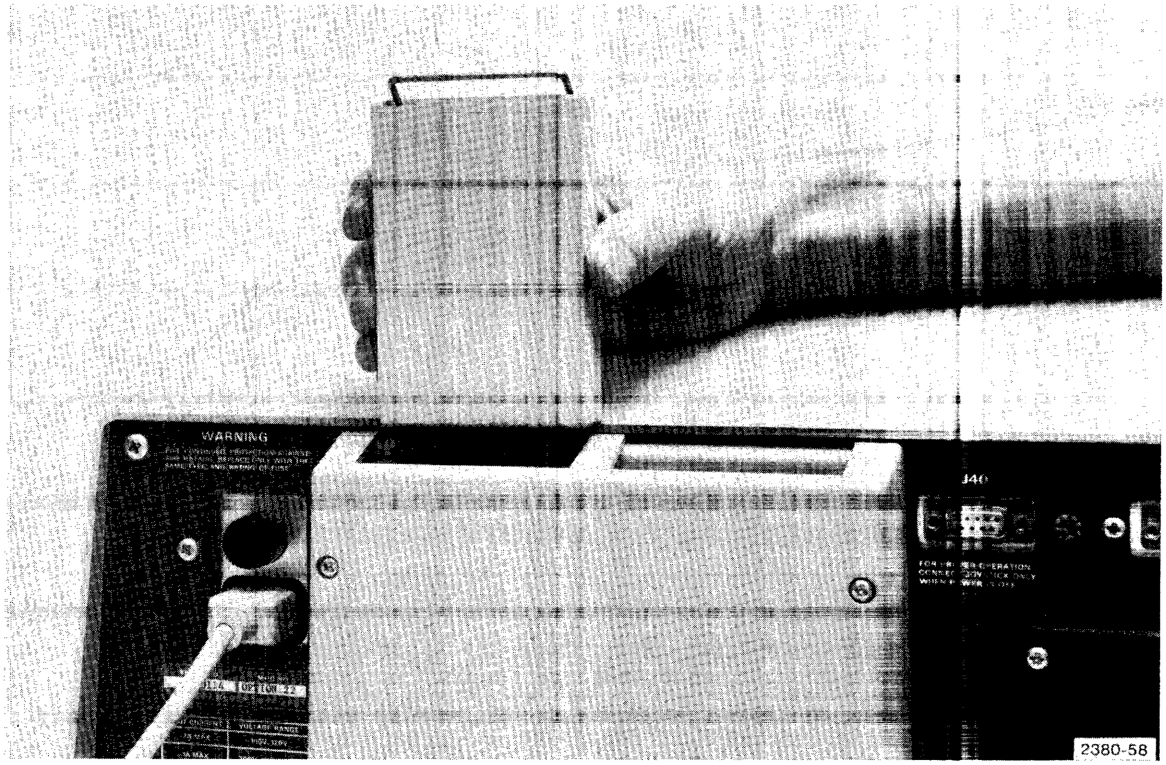


Figure 2-2. Plugging in ROM Pack.

Power Up

CAUTION

If multiple GPIB devices are connected to the Graphic System, at least half of them, plus one, must be turned on before turning on the Graphic System. If this is not done the I/O indicator may light continuously and the system will not operate. The devices must be turned on, even if they are not going to be used, otherwise, they must be disconnected.

4907 (Single Drive)

1. Turn on the front panel power switch on the main 4907 cabinet.

2. Turn on the Graphic System.

4907 Option 30 and 4907 Option 31

1. Turn on the front panel power switch on the auxiliary cabinet.

2. Turn on the front panel power switch on the main 4907 cabinet.

3. Turn on the Graphic System.

Loading the Flexible Disc

1. Press drive door release on front of drive and place flexible disc in cavity as shown (Figure 2-3). Slide disc, with label up, as shown in Figure 2-3, as far in as it will go or until you feel a "click". Be sure the tape supplied with the disc is covering the write-protect hole whenever formatting or writing to a disc (Figure 2-4). To write-protect a disc, remove the tape covering the write-protect hole.

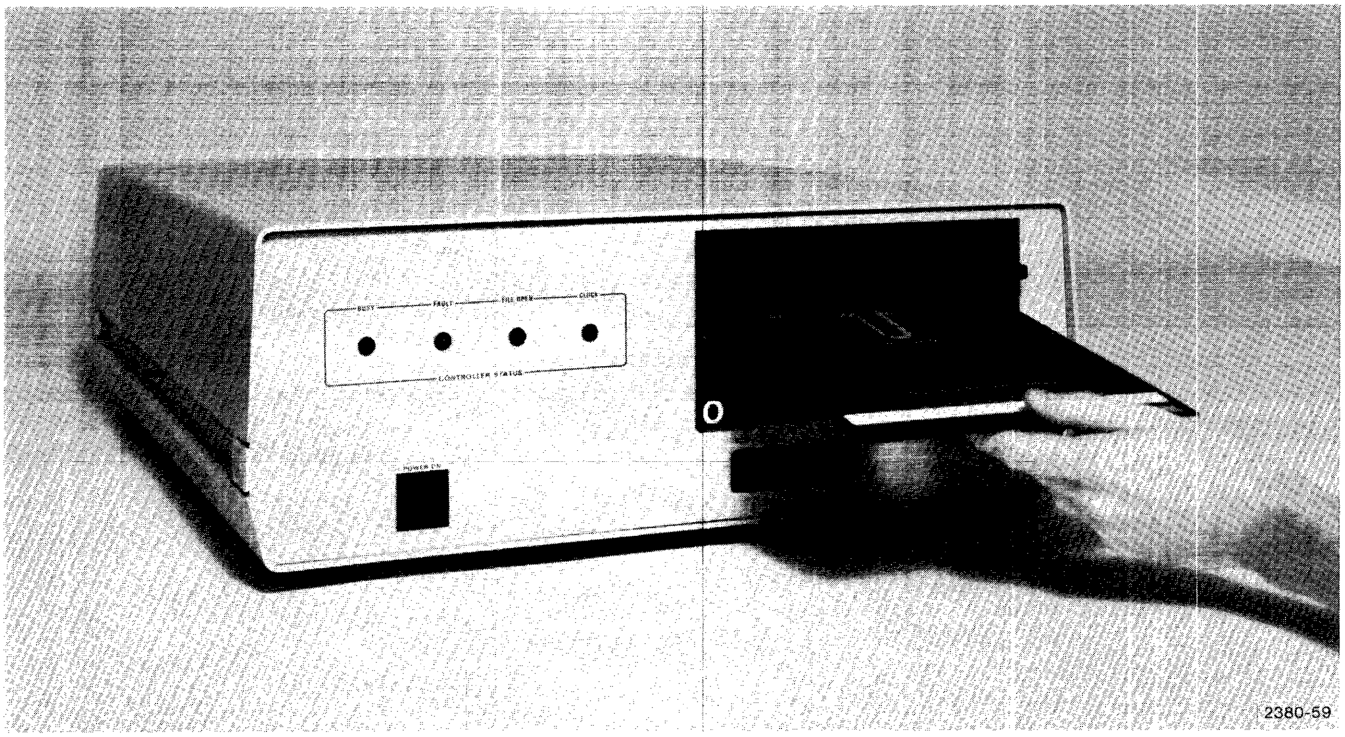
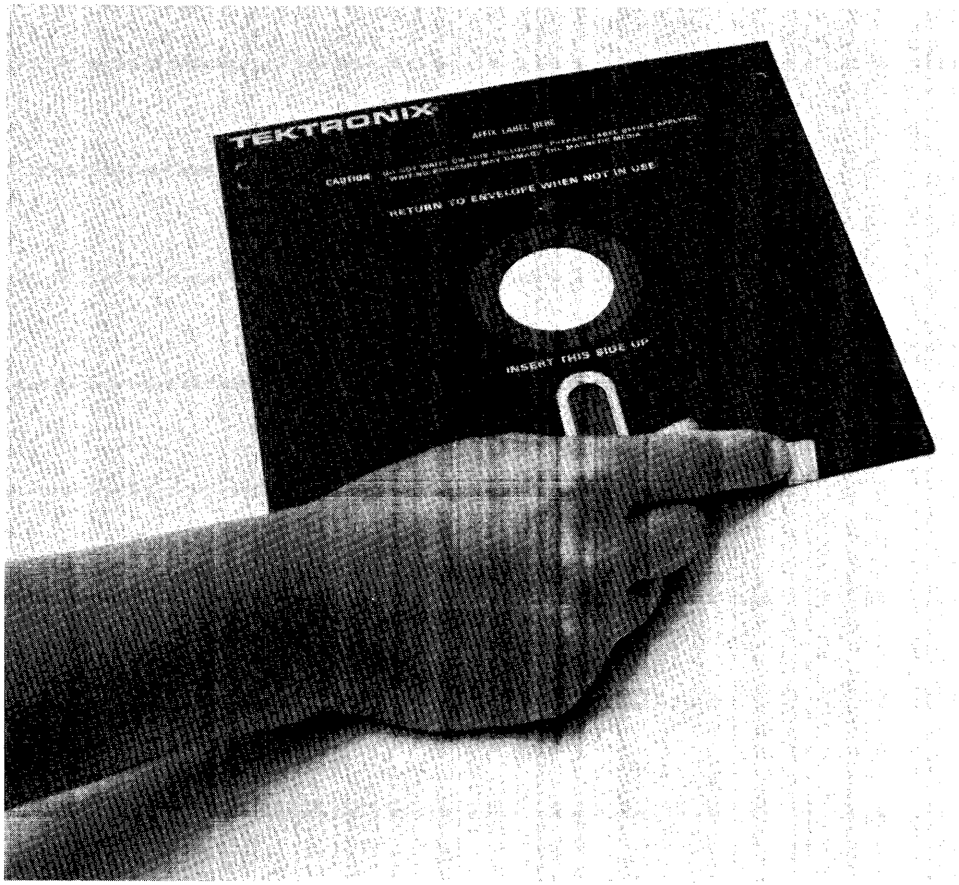


Figure 2-3. Placing Flexible Disc in Drive.



1950-37

Figure 2-4. Flexible Disc Write-Protect Hole.

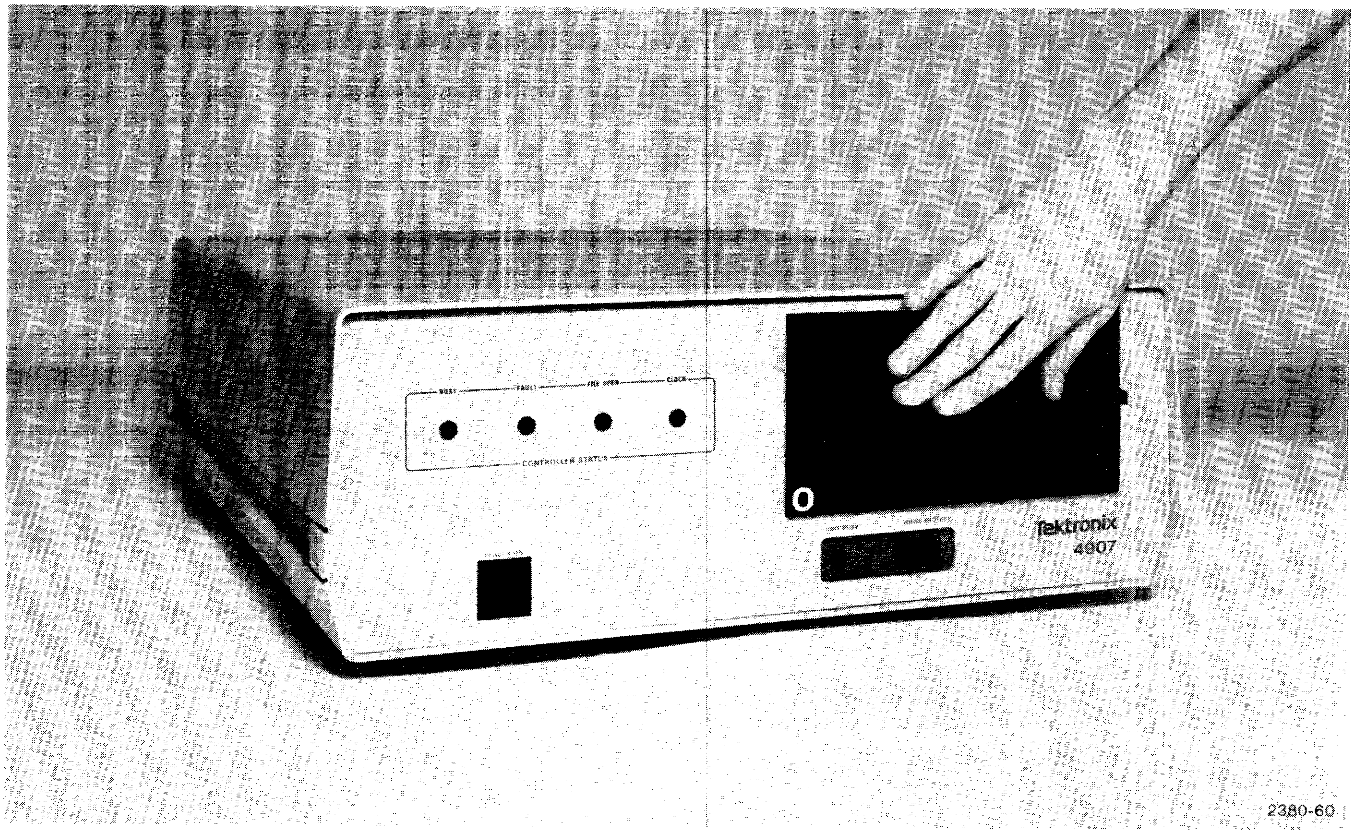


Figure 2-5. Closing Drive Door.

2. Close Door (Figure 2-5).

3. Repeat sequence with additional drives, if necessary.

Flexible Disc Storage and Handling

When using flexible discs, follow these guidelines:

- Do not place heavy objects on disc cartridge.

- Do not touch disc surface. Attempts to clean the disc may damage the surface.

- Do not write directly on cartridge but use labels instead. Use soft-tipped or fiber pens instead of pencils to write on the labels.

- Allow a disc at least five minutes to adjust to existing temperature and humidity before using.
- Store discs vertically in their envelopes to prevent dust from settling on disc surfaces. Storage temperature should be between 10° and 50° C (50° and 120° F).
- Keep cartridges away from heat, sunlight, magnetic fields, and magnetic material. Electromagnetic radiation can destroy stored data.
- Do not use paper clips on a disc.

GENERAL SEQUENCE FLOW CHART

The general sequence flow chart at the back of this manual (Figure 2-6) shows the steps necessary in creating and using files. The accompanying text explains the important details of File Manager operation.

HOW TO GET STARTED ON THE FILE MANAGER

Once the system is powered up, the following steps will get the 4907 up and running. (Be sure to press the RETURN key after each line of text that is entered on the Graphic System keyboard.)

1. LOAD THE DISC.

   Place a blank flexible disc in drive 0. Be sure the write-protect switch is off and the tape is over the write-protect hole on the disc. (See Figures 2-3 and 2-4.)

2. Enter INIT (that is, type INIT on the Graphic System keyboard and press the RETURN key).

   This initializes the system and sets all variables to an undefined state.

3. Enter CALL "SETTIM","DD-MON-YY HH:MM:SS".

   This command, with the current date and time, sets the system clock.

4. Enter DIM A$(200).

   This command increases the size of A$ to allow room for the status message.

5. Enter CALL "DRES",0.

   This command reserves drive 0 for your system.

6. Enter CALL "FORMAT",0,"COMPANY",1,1,"YOUR NAME","MSTRPSWRD",1,10,1,1,1

   This formats the disc in device 0. This means you are giving it its own identification.

   NOTE

   Be sure to write down your password. This password can be used with the KILL and CALL "MRKBRG" commands.

   Formatting a disc takes about two minutes. When formatting is complete, the busy light on the drive goes off.

7. Enter CALL "DREL",0.

   This command releases drive 0.

8. Enter CALL "DSTAT",0,A$.

   This command asks the system to send a message about the disc to A$.

9. Enter A$.

   The device status message appears. A typical message looks like this:

```
4907           DEV ID    COMPANY    VOL ID   YOUR NAME                      OWNER
  630528 FREE       630784 SIZE              0 LOST     256 BLK SIZE
24-AUG-78 09:48 FORMATTED       0 FILES OPEN
```

This message shows the name of the disc (ID) the owner's name, how many bytes are available, etc. See Appendix A for a more complete explanation.

Once your disc has been formatted, the start-up procedure is as follows:

1. Place the formatted disc in drive 0.

2. Set the system clock (if necessary) using the "SETTIM" routine.

3. Dimension A$ to 200 (if necessary). If error message 9 results, A$ is already defined; use a different string name (i.e.,B$).

4. Enter CALL "MOUNT",0,A$ (or B$; whichever you wish to receive the device status message for this disc).

Once a disc is mounted (the FORMAT routine automatically mounts the disc), files can be created on the disc. To store a program on the disc, first load a program into system memory from a tape or enter it from the keyboard; then enter SAVE "FILE". Your program is now stored on the disc in the file named FILE. For program files, you do not need to "mark" the size of the file (as you must for magnetic tape files).

To retrieve the program from the disc, enter OLD "FILE".

SAMPLE I/O

To store data sequentially on the disc, use the following sequence:

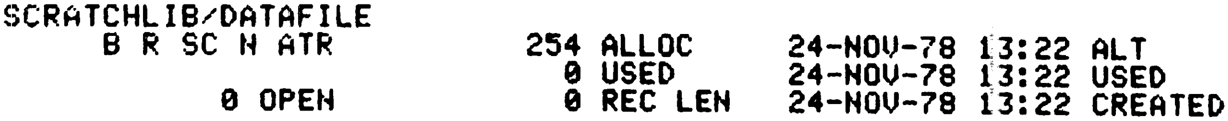
1. Place a formatted disc in drive 0 and mount it.

2. Enter CREATE "DATAFILE"; 200,0.

   This creates a file 200 bytes long on your disc.

3. Enter DIRECTORY 2, "DATAFILE".

   The file status message for the file DATAFILE is displayed (see Appendix A for a more complete explanation):



```
SCRATCHLIB/DATAFILE
     B R SC N ATR             254 ALLOC      24-NOV-78 13:22 ALT
                                0 USED       24-NOV-78 13:22 USED
           0 OPEN               0 REC LEN    24-NOV-78 13:22 CREATED
```

4. Enter Z$="SMITH & JONES 12345" and Z0=12345.

5. Enter OPEN "DATAFILE";1,"F",A$.

   This command opens the file and assigns the logical file number (lfn) 1 to this file.

6. Enter WRITE #1:Z$,Z0.

   This command sends the string Z$ and the variable Z0 in binary format to lfn 1 (values are stored in ASCII format by using the PRINT command).

7. Enter CLOSE 1 to close lfn 1.

   The string "SMITH & JONES 12345" and the number 12345 now reside in the file. To access these values, enter:

   ```
   OPEN "DATAFILE";1,"F",A$
   READ #1:X$,B3
   CLOSE 1
   ```

NOTE

Opening a sequential file for "Full" access positions a pointer at the first data item in that file. As items are retrieved, the pointer points to successive data items. If the file is closed before the pointer reaches the end of the file, all items after the pointer are lost.

The previous example created a binary sequential data file. Data files can be either random or sequential. This example creates an ASCII random file.

1. Enter CREATE "ASCIRNDM";"A";20,31.

   This command generates an ASCII file named ASCIRNDM which consists of 20 records (where each record is 31 bytes long). See FILE OR FILE RECORD SIZE REQUIREMENTS under the CREATE command description for a discussion of spare requirements.

In order to store data randomly to a random file, the file must first be "initialized". Initialization in this case means filling each record with blanks.

2. Enter OPEN "ASCIRNDM";1,"F",A$.

   This command opens the file and assigns lfn 1 to it.

3. Enter Z$=" ", inserting 30 blanks between the quotation marks.

   To verify that Z$ contains 30 blanks, enter LEN(Z$). The value returned equals the number of characters in Z$.

4. Enter PRINT #1,1:Z$.

   This command stores the contents of Z$ in record 1 of lfn 1.

5. Enter PRINT #1,2:Z$.

   This command stores the contents of Z$ record 2 of lfn 1.

Enter a print statement for all 20 records of ASCIRNDM (for example, with a FOR ... NEXT loop).

To store numeric or string data in ASCIRNDM, enter:

    PRINT #1,I:<variable or variable name>

where I is a record number.

To retrieve data, enter:

    INPUT #1,I:<variable name>

where I is the record number. If the named variable is X$, the data string is stored in X$.

NOTE

For ASCII random files, even though they must be initialized, the End of Record characters are ASCII characters. It is therefore possible to read past the end of a record (i.e., if there are more variable names in your INPUT statement than there are data items in that record). End of Record characters will not transfer control to ON EOF ... THEN ... statements.

The following program shows the basic steps necessary to create files and write and print to those files. It also shows how to retrieve and print the information stored in those files.

If error messages occur, check ERROR MESSAGES AND RECOVERY PROCEDURES (Appendix A). Be sure that all entries are correct and that all preparatory steps described earlier in this section have been performed.

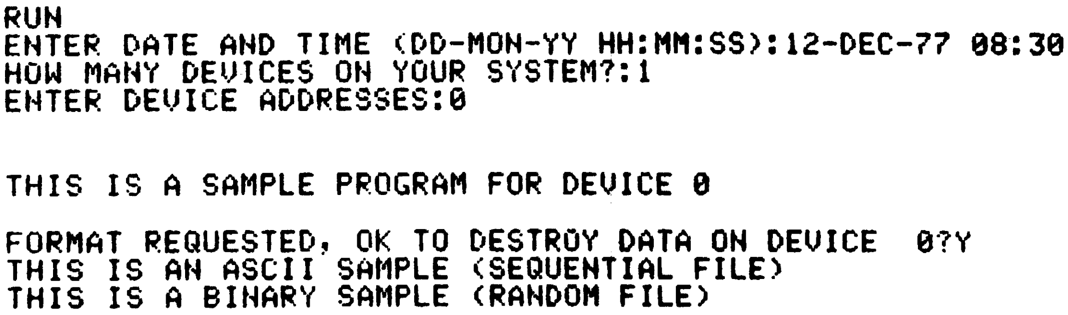
A description of the program operations and a sample of the program output follow the program listing below.

```
100 INIT
110 DIM R$(2000),F$(200)
120 CALL "TIME",R$
130 IF LEN(R$)>0 THEN 170
140 PRINT "ENTER DATE AND TIME (DD-MON-YY HH:MM:SS):";
150 INPUT A$
160 CALL "SETTIM",A$
170 PRINT "HOW MANY DEVICES ON YOUR SYSTEM?:";
180 INPUT N
190 DIM D(N)
200 PRINT "ENTER DEVICE ADDRESSES:";
210 INPUT D
220 FOR I=1 TO N
230 PRINT "J̲J̲THIS IS A SAMPLE PROGRAM FOR DEVICE ";D(I);"J̲"
240 CALL "UNIT",D(I)
250 CALL "DRES",D(I)
260 CALL "FORMAT",D(I),"SAMPLE",1,1,"OWNER","PASS",1,10,1,1,1
270 CALL "DREL",D(I)
280 CREATE "ASCFILE","A";1,0
290 CREATE "BINFILE";1,128
300 OPEN "ASCFILE";1,"F",F$
310 PRINT #1:"THIS IS AN ASCII SAMPLE (SEQUENTIAL FILE)"
320 OPEN "BINFILE";2,"F",F$
330 WRITE #2:"THIS IS A BINARY SAMPLE (RANDOM FILE)"
340 CALL "REWIND",1
350 INPUT #1:S$
360 PRINT S$
370 READ #2,1:S$
380 PRINT S$
390 CLOSE
400 NEXT I
410 END
```

Program Description

100	Initialize
110	
120	
130	
140	
150	Set system clock, if necessary
160	
170	
180	Enter the total devices on your system (1, 2, or 3)
190	
200	
210	Enter the device addresses (0 or 0, 1, etc.)
220	
230	Print heading
240	Set unit number
250	Format disc
260	
270	
280	Create an ASCII, sequential file
290	Create a binary, random file
300	Open each file and store message
310	
320	
330	
340	Rewind sequential file
350	Access files and display messages
360	
370	
380	
390	Close both files
400	
410	End program

Sample Program Output



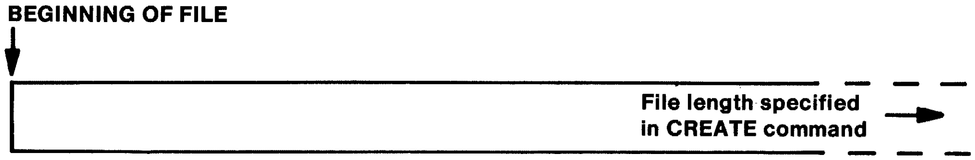
```
RUN
ENTER DATE AND TIME (DD-MON-YY HH:MM:SS):12-DEC-77 08:30
HOW MANY DEVICES ON YOUR SYSTEM?:1
ENTER DEVICE ADDRESSES:0

THIS IS A SAMPLE PROGRAM FOR DEVICE 0

FORMAT REQUESTED, OK TO DESTROY DATA ON DEVICE  0?Y
THIS IS AN ASCII SAMPLE (SEQUENTIAL FILE)
THIS IS A BINARY SAMPLE (RANDOM FILE)
```

RANDOM AND SEQUENTIAL FILES

All 4907 File Manager files cover specific areas of specific length on the disc recording surface. The location and size is specified in the CREATE command. There are two basic file forms — sequential and random.

Sequential Files

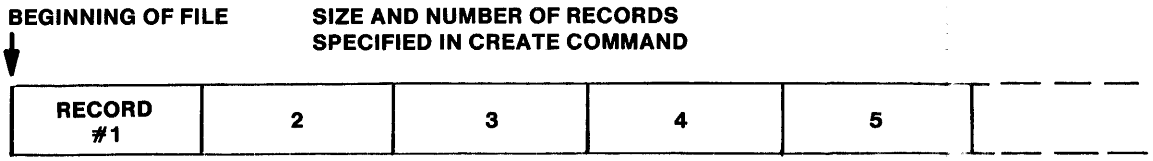
A sequential file may be pictured as a long empty space with no divisions:



Information is entered sequentially. This means it must start at the first empty space and work towards the end of allocated space. The first empty space may be at the beginning of the file or just past the last data entered. Information retrieval also must be sequential and must start at the beginning of the file. Sequential files are useful in storing entire programs or data items that are best accessed one after another. Sequential files may be generated on discs and Graphic System magnetic tapes.

Random Files

A random file is similar to a sequential file except that it is divided into two or more section or "records."



The size and number of these records are specified in a CREATE command. These records, which may be as small as one byte in length, are numbered. Information may be entered into or retrieved from any record in the file by specifying that record number in the I/O command. This means that data as small as a single character or number may be written or read or printed or input immediately without a sequential search. Data in each record cannot be "added to." Each time new information is entered into a record, it completely replaces existing information. Information in all other records in the file, however, remains unchanged.

Random files may be created only on 4907 File Manager discs. They cannot be generated on the Graphic System internal magnetic tape.

NOTE

Files are specified random or sequential in the CREATE command.

SPECIAL SYSTEM FEATURES

Special 4907 File Manager features are listed below with the associated commands. See COMMAND DESCRIPTIONS (Section 5) for specific command details.

System Clock

When files are created, altered, or used, the date and time of the activity are automatically recorded for that file.

The date and time of the last disc formatting are recorded on the volume label.

Command involved:

```
CALL "SETTIM"
```

Automatic File Extending

Whenever data is sent to a file requiring more than the allocated space, the system automatically extends the file to accommodate the extra data.

Commands involved:

```
WRITE
PRINT
SAVE
```

Status Messages

Whenever it is necessary to check the status of a file or device, special commands may be executed which return status messages. These messages include time/date information, space specification, identification, etc. See DEVICE AND FILE STATUS MESSAGES (Appendix A).

Commands Involved:

```
CALL "CUSTAT"
CALL "FILE"
CALL "MOUNT"
CALL "DSTAT"
DIRECTORY
OPEN
```

Special Characters

Special characters in file identifiers simplify writing F.I.s and the search for files or groups of files. They are also useful in accessing certain kinds of libraries. See SPECIAL CHARACTERS IN FILE IDENTIFIERS in Section 4.

"Group" Open/Next File

Sometimes you may need to access several files consecutively. Normally, this would require a separate OPEN for each file. However, with the use of SPECIAL CHARACTERS, groups of files may be selected, opened, and used with a single OPEN. This kind of OPEN command is called a "GROUP" OPEN and requires that a "G" is entered in the command. After the first file in the group is opened and used, CALL "NEXT" closes that file and opens remaining files one by one.

Commands involved:

OPEN
CALL "NEXT"

Free Space Message

The amount of remaining storage space on any disc (in bytes) may be seen by requesting a device status message.

Commands involved:

CALL "CUSTAT"
CALL "DSTAT"
CALL "MOUNT"

Random Access

Unlike files on magnetic tape, disc files may be created for random or "direct" access. This means that the file may be divided into numbered "sections" or records as small as one byte. By entering the number of that record in an I/O command it can be directly accessed without the need of stepping through the data items one by one. This can greatly speed I/O activities. You can specify the disc files to be random or sequential in the CREATE command.

Simultaneous File Use

Once the system is notified which drive is to be accessed, up to nine files may be opened and used. See OPENING MULTIPLE FILES in the OPEN command description.

Commands involved:

UNIT
CALL "UNIT"

SECTION 3

CONTENTS

Section 3

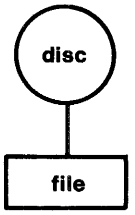
STORAGE STRUCTURE

INTRODUCTION

This section describes the concepts of file storage structures. These concepts are helpful when you begin to write file identifiers (F.I.s) which are a part of many commands.

WHAT IS A STORAGE STRUCTURE?

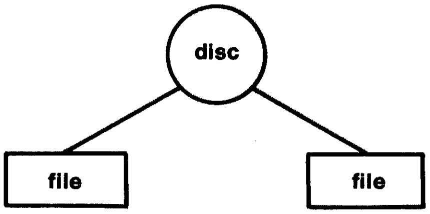
When data or program information is sent to a disc, it is stored in a "file" that occupies a portion of the disc. Usually each file contains different information. Often, however, several files may contain similar information. As a result, it may be desirable to group files into categories, and then collect the categories into groups, and so on. This way it is easier to keep track of and recover all information. When one or more files are placed on the disc, or when file grouping is carried out, a "storage structure" is formed. To illustrate what "storage structure" is and why it is a useful concept, Figure 3-1 shows a "storage structure" containing a single file.



2380-14

Figure 3-1. Sample Storage Structure (1 Level).

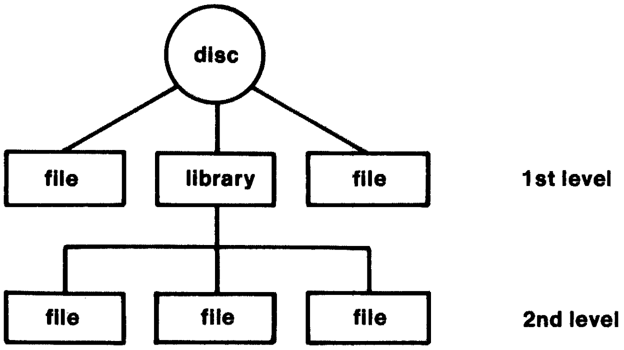
If another file is added, the structure in Figure 3-2 results.



2380-15

Figure 3-2. Sample Storage Structure (1 Level).

So far there are two files stored on the "1st level". The 4907 File Manager System provides five storage levels. If several files sharing a common characteristic or belonging to a particular user are added, they may be grouped under a "LIBRARY." If a library is used, the files will be placed on the next level; for example, three files grouped in a 1st level library will be placed on the 2nd level as shown in Figure 3-3.



2380-16

Figure 3-3. Sample Storage Structure (2 Levels).

A library is a "heading" and can only be used to reference other libraries or files on lower levels. See the CREATE command to see how libraries are specified. A library can never be used to hold data or programs, nor can a file ever reference another file or library. For example, if two groups of files were stored on the 5th level, there must be preceding libraries on the 4th, 3rd, 2nd and 1st levels. The storage structure in Figure 3-4 shows what this might look like if these 5th level files were added to the sample storage structure in Figure 3-3.

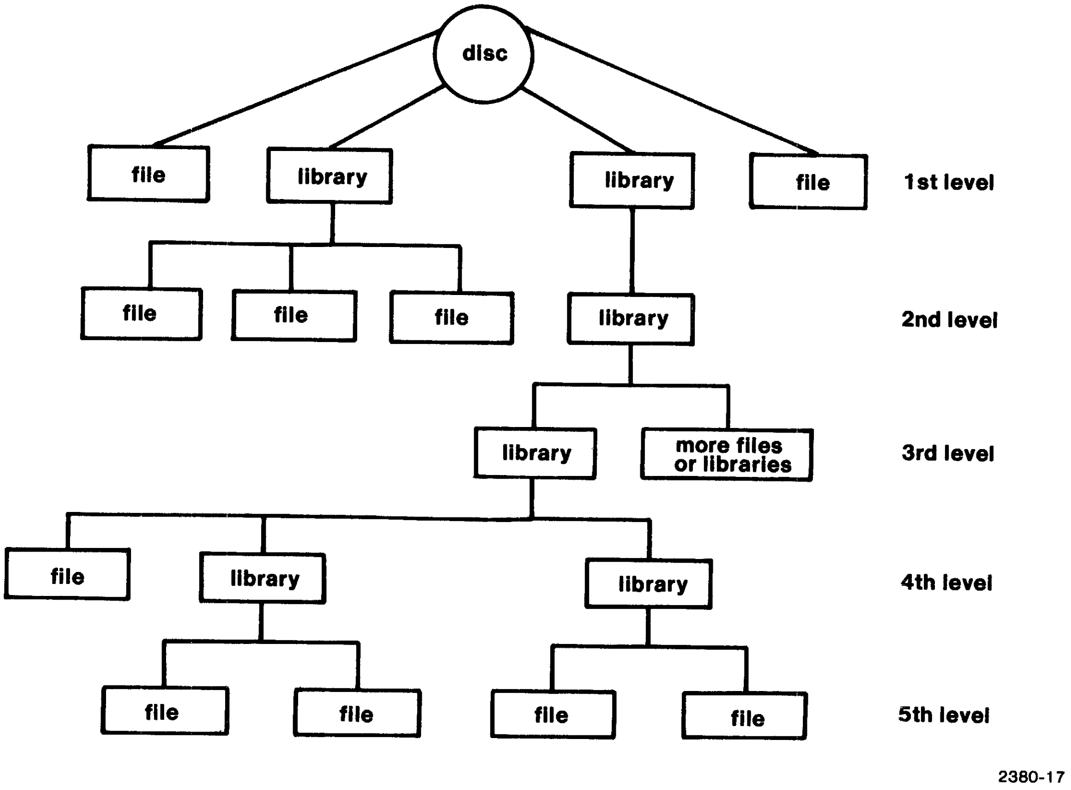


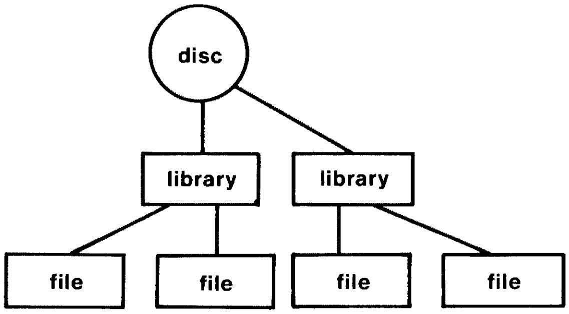
Figure 3-4. Sample Storage Structure (5 Levels).

Remember:

- The only purpose of libraries is to allow grouping of files or libraries on subsequent levels. As a result, no libraries may be placed on the 5th level.
- Files never have subsequent files or libraries.
- There is no limit (other than space restrictions) to the number of libraries or files on a particular level.
- Storage structures are limited to five levels.

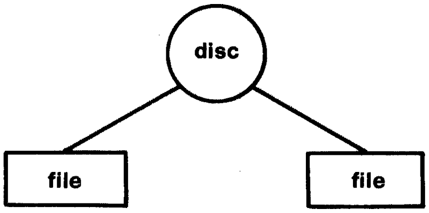
NOTE

When separate discs are dedicated to an individual subject or user, storage structures can be very simple because fewer storage levels are necessary. For example, a single disc with two libraries (ignoring the standard first level libraries: SYSLIB, USERLIB and SCRATCHLIB) containing two files each will have a storage structure like this:



2380-63

Three discs with only two files each have a much simpler structure:



2380-15

SECTION 4

CONTENTS

Section 4 **HOW TO WRITE A COMMAND** Page

Section 4

HOW TO WRITE A COMMAND

SYNTAX

4907 File Manager commands, like other Graphic System BASIC commands, must be written in a specific manner. The fields required for each command must be in proper order, and the information entered in those fields must be of the type specified in the description. Generally, each command contains one or more of the following fields in the order shown:

KEYWORD (or keywords)	and ADDRESS (or addresses)	and/or ARGUMENT

Keyword

The command keyword tells the system what kind of activity is to be carried out. The entry may be abbreviated as shown in the description of that command. Two keywords are required for some commands. Some keywords are preceded by the word CALL. Keywords preceded by CALL must always be in quotes; for example, CALL "DSTAT".

Address

The address tells the system the location of the device, the location of the file, or both. In some cases, two addresses are necessary because two devices or files are needed. The different types of addresses required are as follows:

device address — This number, which must be an integer from 0 to 255, tells the system which device it will be communicating with.

F.I. — File identifier. This combination of names tells the system the location and identity of the file or files on the disc.

HOW TO WRITE A FILE IDENTIFIER (later in this section) and STORAGE STRUCTURE (Section 3) provide discussion of F.I. requirements and storage structure.

lfn

Logical file number. The integer, from 1 to 9, entered in this field is specified in an earlier OPEN command and represents BOTH the device address and the F.I. The main purpose of an lfn is to make it unnecessary to include both the device address and the F.I. in subsequent commands. In some commands the lfn must be preceded by # (pound sign) as shown in the description of that command.

I/O address

The I/O address tells the system what device, other than the Graphic System display, is to receive file status messages and what action is required. An I/O address may be specified only in the DIRECTORY command.

Only the CALL "SETTIM", DELETE ALL, CALL "CUSTAT", CALL "FMVALS", SECRET, CLOSE, and CALL "TIME" commands do not require an address of some kind. This is because they set or reflect parameters affecting the entire system.

Argument

Those fields not containing keywords or addresses are considered "argument" fields. The argument may tell the system where to send the information that the command is generating, as in this example:

```
CALL "DSTAT",2,A$
```

The last field, or argument, tells the system to send the device status message that this command generates to A$ in the Graphic System memory.

The argument may be used to supply information for command execution rather than to store the results. The "2" in the preceding command specifies drive 2.

Some commands, such as CALL "FORMAT", have many argument fields; others, such as CLOSE, may have none.

CLASSES OF FIELDS

As you read the syntax forms in COMMAND DESCRIPTIONS (Section 5), you will see the fields are specified as one or more classes. An entry in any field must fit one of those classes. Each class is described below.

Constant

A constant entry must be a positive integer and cannot be represented by a variable.

String Constant

A string constant entry must be the string required for that field and CANNOT be represented by a string variable (such as A$). All string constants must be placed in quotation marks ("DOCTORX") when entered in a command.[1] In some cases, you must make a choice between a few single character constants. Those choices will be shown instead of the term "string constant" within brackets, like this:

$$\left\{\begin{array}{c}\text{"A"}\\ \text{"B"}\\ \text{"C"}\end{array}\right\}$$

Numeric Expression

A numeric expression may be any of the following:

Numeric constant, such as 3

Numeric variable, such as M

Subscripted array, variable such as B(1),B(1,1)

Logical or relational comparison enclosed in parenthesis, such as,

(A or B)
(A > B)

Valid combination of the above

[1] When a series of string constants are used in an F.I., only a single set of quotation marks around the entire F.I. is necessary.

Numeric Variable

The entry may be any numeric variable allowed in regular Graphic System operation. If a numeric variable is entered, it must have a value already defined.

String

The entry may be a string ("DOG") or a string variable representing that string (A$). Any string must be preceded and followed by quotation marks when entered in either the command or stored in a string variable as shown here:

```
A$ = "FINANCE"
```

String Variable

The entry may be any valid string variable allowed in regular Graphic System operation (e.g., A$, B$).

All string variables must be dimensioned if more than 72 characters are required to contain the information generated by the command. See the 4050 Series Graphic System Reference Manual.

Target String Variable

Usually, when string variables are used in a command, they already contain data to be used by that command — but not always. In some command descriptions you will see the term "TARGET STRING VARIABLE." In these cases the string variable is redefined at the time of command execution, and the data generated by the command is sent to this target string variable.

The following command example is written with an integer, a string and, in the last field, a target string variable:

```
CALL "FILE",3,"MYLIBRY/FILE",A$
```

The target string variable (A$) is the destination of the message generated by this command.

If the target string variable already contains data, it is replaced by the new data when the command is executed.

Target Numeric Variable

The numeric result of a command is sent to a target numeric variable.

COMMAND DELIMITERS, PUNCTUATION AND SPACES

Special delimiters are required in F.I.s. See DELIMITERS later in this section for details on how they are used.

Commas, Colons and Semicolons ,:;

Separate fields. Their exact use is shown with individual command descriptions.

Quotation Marks " "

Tell the system that the enclosed data is a string constant which may be a single character or numeral, a string of alphanumerics, or a blank indicating a null string.

Spaces

Generally, spaces are ignored. Occasionally, however, they must be used as delimiters; for example, the system will return an error message if a SAVE command is written like this:

```
SAVE$;100,500
```

The syntax form of each command shows when spaces are required.

The system has interpreted the E in E$ as part of the keyword SAVE. The command must be written with a space, like this:

```
SAV E$;100,500
```

Trailing Dots ...

If a syntax or descriptive form is followed by trailing dots, the preceding field may be repeated as many times as desired.

NOTE

See SYNTAX in the 4050 Series Graphic System Reference Manual for further details.

HOW TO WRITE A FILE IDENTIFIER

STORAGE STRUCTURE (Section 3) describes libraries and files. This section explains how to use a File Identifier, abbreviated F.I., to create and access those libraries and files.

Whether the F.I. is used to create a new file or locate an existing file depends on the command in which it is placed. The F.I. is useless by itself. It (or a logical file number (lfn) representing the F.I.) must go inside a command.

Every file has its own F.I.. No two files have exactly the same F.I. unless they are on different discs.

What is a File Identifier?

A file identifier is the name of a file in a format the File Manager can understand. The syntax form of this format is as follows:

```
[@][[[[ library name/]]]] {file name}[.extension][:password]
```

Field Descriptions

@ (The Commercial "at" Sign)

> The "@" is used to circumvent the "current library." By using the @ the system will access any library entered. See USERLIB String Suppression in this section and the CALL "USERLIB" command description in Section 5.

library name

> The entry here specifies the name of the library. For convenience, first level libraries fall into groups named SYSLIB, USERLIB, or SCRATCHLIB. Each group is accessed differently. See USE OF 1ST LEVEL LIBRARIES in this section.

file name

> The entry here specifies the name of the file. The character requirements are the same as for libraries.

Extensions

The extension allows the program to distinguish between similar, but not identical, files by acting as a label or tag on the end of the name of the file.

EXAMPLE:

File: VEG

Files with extensions: VEG.PER or VEG.ANN

An extension may be specified for any file, but an extension is never assigned to a library. The extension may be 1 to 4 characters long. The first character must be alphabetic, the rest alphanumeric.

The sample storage structure in Figure 4-1 and the description following it show how extensions are used:

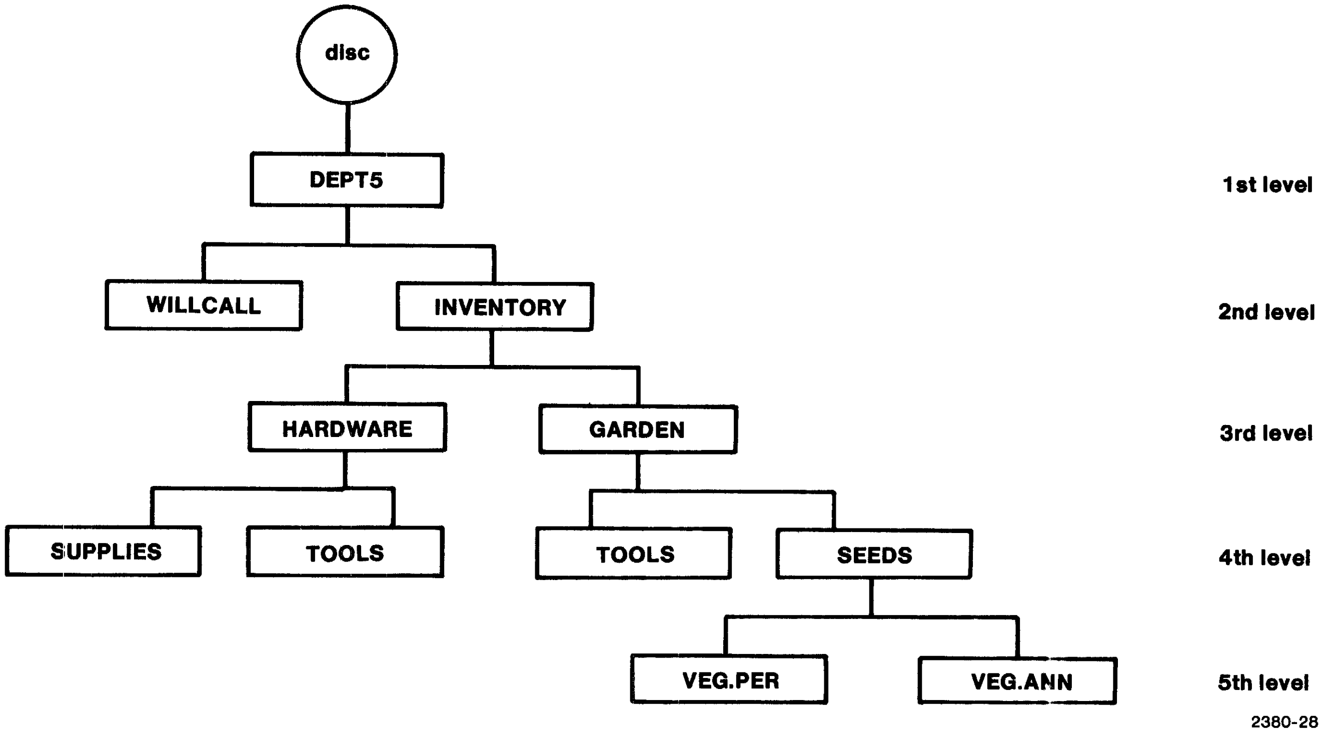


Figure 4-1. Sample Storage Structure (5 Levels, 6 Files).

This structure shown in Figure 4-1 includes files WILLCALL, SUPPLIES, TOOLS, TOOLS, VEG.PER and VEG.ANN. All the other entries are libraries. Since there are six files in this structure, six F.I. s were required to construct it. Notice the extensions "PE" and "ANN" were added onto the "VEG" files. This is to allow the Graphic System to distingush between them. You may have noticed that no extensions were specified for the files named "TOOLS". This is because they are under different libraries. Once the Graphic System reaches either "HARDWARE" or "GARDEN" in its search for a "TOOLS" file it is unaware that there is another "TOOLS" file anywhere else. As a result, no extensions are necessary.

Passwords

Passwords may be assigned to any library or file to prevent unauthorized access. When attempts are made to access the file, the password or passwords must be entered in the F.I. with the file and library names, or access will be prevented or restricted.[2]

NOTE

Be sure to write down passwords when they are assigned.

Passwords are specified in F.I. s along with library names, file names and extensions. The password must immediately follow the name of the library or the name of the file, and must be separated by a colon (:). The format is shown below:

library or file name:password

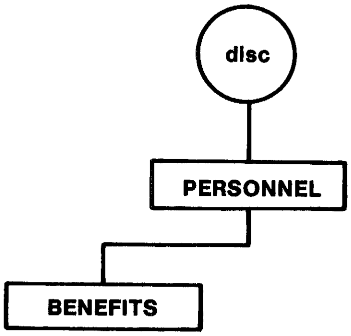
A password may be 1 to 10 characters long. The first character must be alphabetic, the rest alphanumeric.

Assume that a new file name "SALARIED" along with the password "BOSS" is to be created in an existing storage structure under a 1st level library "PERSONNEL." The F.I. in the CREATE command must be written this way:

"@PERSONNEL/SALARIED:BOSS"

Suppose the existing storage structure looks like Figure 4-2.

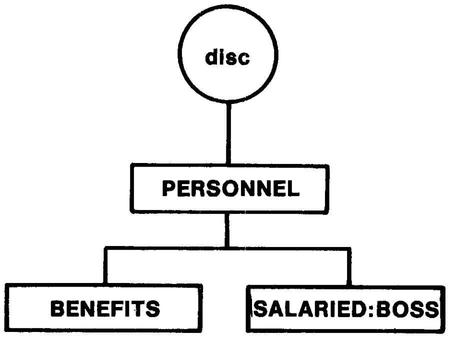
[2] Unless a master password is used. See CALL "FORMAT" command description in Section 5.



2380-25

Figure 4-2. Sample Storage Structure (2 Levels, 1 File).

A CREATE command with this F.I. results in the revised storage structure shown in Figure 4-3.



2380-26

Figure 4-3. Sample Storage Structure (2 Levels, 2 Files, 1 With Password)

Remember that:

- A password cannot be assigned to the "PERSONNEL" library because it existed before this new F.I. was written.
- A file name password can be changed by executing a CALL "RENAME" command if the original password is specified in the command. Library passwords cannot be changed.

NOTE

When separate discs are dedicated to an individual user or subject the F.I.s can be very simple. This is because fewer storage levels are necessary.

For example:

To access a file on a disc with two libraries (ignoring the standard first level library (see "Use of 1st Level Libraries", later in this section)), each containing two files, requires an F.I. like this:

"library/file"

Accessing a file on a disc exclusively devoted to files requires an F.I. like this:

"file"

By dedicating a disc to a particular user and his own program, the disc can always remain in his possession. This means passwords may be eliminated altogether.

Use of 1st Level Libraries

There are three types of 1st level libraries:

SYSTEM LIBRARY	USER LIBRARY	SCRATCH LIBRARY
(SYSLIB)	(USERLIB)	(SCRATCHLIB)

SYSLIB: This library contains programs for system control, such as data acquisition. It also may be used for commonly accessed public programs for math or statistics work. There is only one system library, but it can be as large as an entire disc.

USERLIB: These libraries contain files usually restricted to a particular user, public files for a variety of users, or a combination of the two. There is no limit to the number of user libraries a user may create, and each may be as large as an entire disc. Every user library created must have a different name. For example: MYLIBRY, DOG, etc.

SCRATCHLIB: This library contains information, sample programs, etc. This is the default library. Unless directed otherwise, the system will always access this library for file creation or selection. There is only one scratch library but it may be as large as an entire disc.

Each type of 1st level library may be addressed as shown below:

SYSLIB: Enter the $ followed by the rest of the libraries and file names. Example: "$A/B/C" (which is the same as "SYSLIB/A/B/C"). This will suppress the current library.

USERLIB: Enter "@" then the name of the user library then the rest of the library and filenames.

Example: "@MYLIBRY/A/B/C" will suppress the current library and let the system access MYLIBRY .

SCRATCHLIB: Leave out the name of the 1st level library, SCRATCHLIB, then enter the rest of the library and filenames.

Example: "A/B/C".

The system will always default to the scratch library if there is no other current library. If there is a current library, it must be suppressed this way:

"@SCRATCHLIB/A/B/C"

NOTE

It is important to remember that these groups are for convenience only. Any or all data may be placed in any one of the above categories.

Examples

The following examples illustrate how to write F.I.'s for creating or locating files in SYSLIB, USERLIB and SCRATCHLIB storage structures ("storage structure" means all libraries and files under these 1st level libraries). To see how characters are used as delimiters in F.I. construction, see DELIMITERS in this section.

SYSLIB

Examples 1 through 3 show how F.I.'s should be written for the "SYSLIB" storage structure (Figure 4-4). These F.I.'s circumvent any current library. This means the system will not look in the current library for the file.

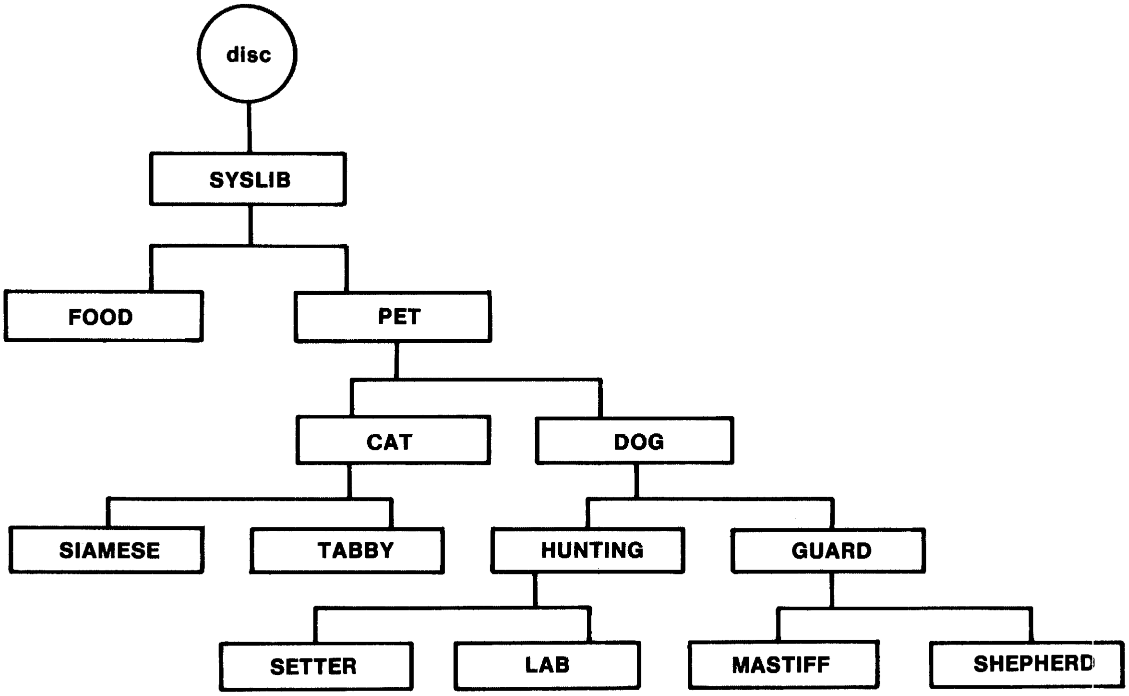


Figure 4-4. Sample Storage Structure (5 Levels, 7 Files).

Example 1:

"$PET/DOG/HUNTING/LAB"

This example locates SYSLIB (by using the $) then libraries "PET," "DOG" and "HUNTING" in succession. The file "LAB" is accessed last.

Example 2:

"$FOOD"

This example locates the file "FOOD" in "SYSLIB."

Example 3:

A$

If the statement

A$ = "$FOOD"

is executed previously, the same file as in example 2 will be located if A$ is entered in the F.I. field. Although this file is on the 2nd level, it has no subsequent files and is, therefore, a "lowest level" file.

USERLIB

The following examples show how F.I.'s should be written for the "MYLIBRY" storage structure in Figure 4-5.

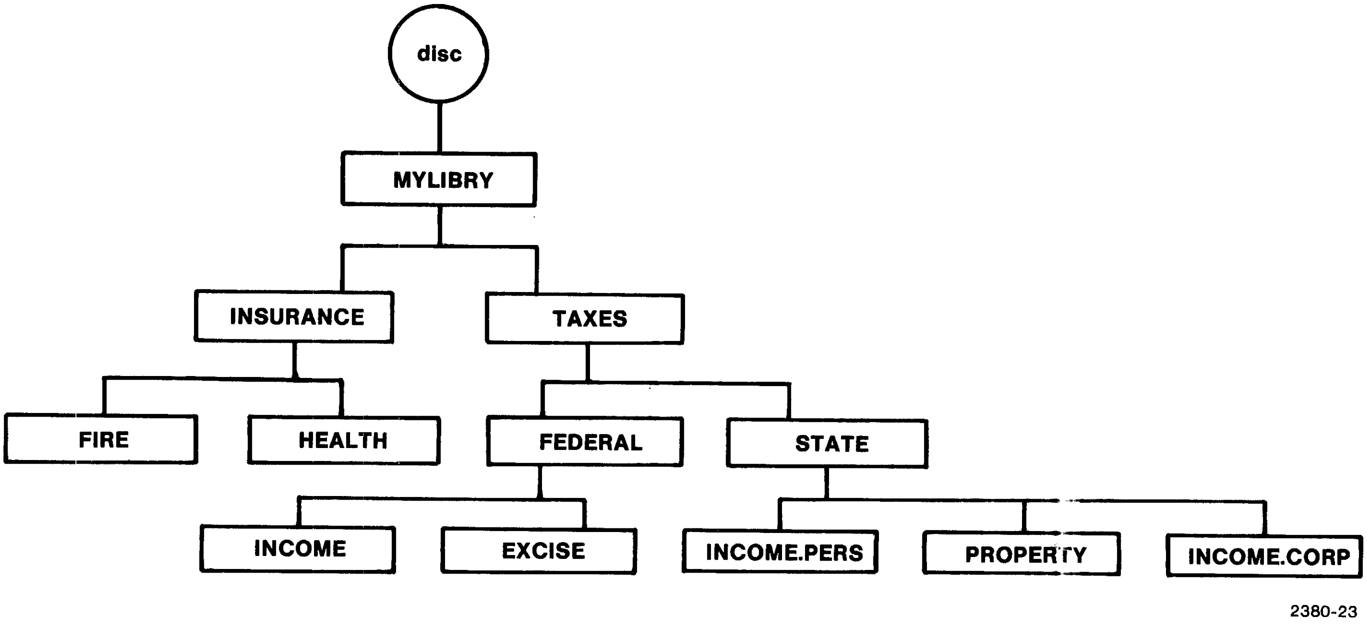


Figure 4-5. Sample Storage Structure (4 Levels, 7 Files).

Examples 4 and 5 show how F.I.'s are written to circumvent the current library. Examples 6 and 7 show how to write F.I.'s when "MYLIBRY" is the current library.

Example 4:

```
"@MYLIBRY/TAXES/FEDERAL/EXCISE"
```

This example specifies the file "EXCISE" in libraries "MYLIBRY," "TAXES," and "FEDERAL." The @ is necessary to circumvent the current library.

Example 5:

```
"@MYLIBRY/TAXES/STATE/INCOME.CORP"
```

This example specifies the file "INCOME" (with the extension ".CORP") located in libraries "MYLIBRY," "TAXES," and "STATE." The @ is necessary to circumvent the current library.

Example 6:

```
CALL "USERLIB", "@MYLIBRY"
"INSURANCE/FIRE"
```

This example specifies the 2nd level library "INSURANCE" in "MYLIBRY"; then searches for the file "FIRE." No 1st level entry is necessary because MYLIBRY was specified in the CALL "USERLIB" command as the current library.

Example 7:

```
C$
```

If the statement

```
C$ = "INSURANCE/FIRE"
```

is executed earlier, C$ will select the same file as Example 6 when entered into the F.I. field in the command.

SCRATCHLIB

The following examples show how an F.I. must be written for the sample scratch library in Figure 4-6.

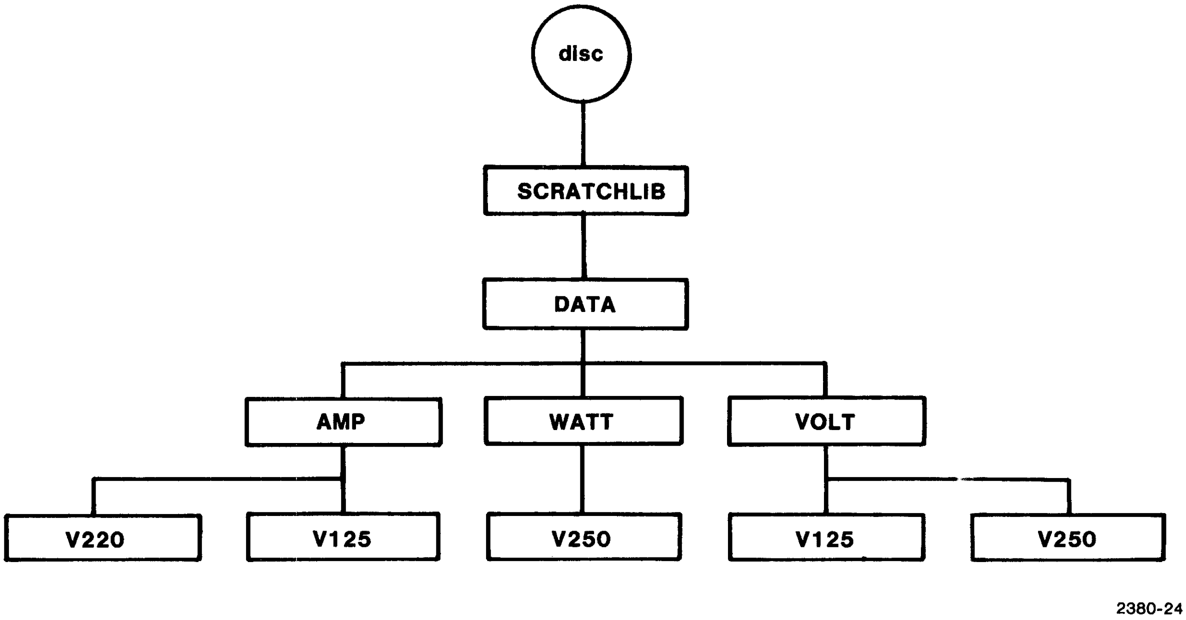


Figure 4-6. Sample Storage Structure (4 Levels, 5 Files).

Example 8 shows how an F.I. must be written to access a file in SCRATCHLIB (the default library).

Example 9 shows how to write an F.I. to SCRATCHLIB when there is a current library.

Example 8:

"DATA/VOLT/V250"

This example locates file "V250" in libraries "DATA" and "VOLT" in the current default library "SCRATCHLIB."

Example 9:

"@SCRATCHLIB/DATA/AMP/V125"

In this example, the first entry (@SCRATCHLIB) circumvents the current library and sends the system to the scratch library. The file "V125" is then located. This example shows how to locate the scratch library when it is not the current library.

Delimiters

The slash (/) is used to separate name fields.

EXAMPLE:

```
@MYLIBRY/A/B/C/D
```

The colon (:) is only used to separate a password from a library name or file name.

EXAMPLE:

```
@MYLIBRY/A/B/C/D:PASS
```

The period (.) is only used to separate an extension from the rest of the file name.

EXAMPLE:

```
@MYLIBRY/A/B/C/D:PASS.EXT
```

OR

```
@MYLIBRY/A/B/C/D.EXT
```

Quotation marks ("") are used to enclose the entire F.I.

EXAMPLE:

```
"@MYLIBRY/A/B/C/D.EXT"
```

No spaces may be included in F.I.'s.

No delimiters can be used in an F.I. except those specified here.

Special Characters #, *, ?,@ , $

The special characters are the pound sign (#), the asterisk (*), the question mark (?), the commercial "at" sign (@), and the dollar sign ($).

Special or Multiple File Selection (#)(*)(?)

The following commands can be written to locate more than just a single file: OPEN, DIRECTORY, KILL, COPY ... TO, CALL "RENAME," CALL "FILE". By using special characters you can write commands that will find:

- All files on all levels below any 1st level library.
- All files on any single level.
- All files with common names, common prefixes or common extensions on any single level.

The files involved will be chosen in the order they are listed in the directory. Only "eligible" files will be selected, and files are eligible only if:

- Passwords are entered completely and correctly.
- Device or disc is not write protected.
- The MOUNT command has been executed.

Pound Sign (#)

The pound sign selects all eligible libraries and files at the level the character is entered as well as all libraries and files at subsequent levels.

Limitations:

- Only one "#" may be used in any F.I.
- Slash marks (/) cannot be used immediately adjacent to a "#".
- Only one file level (with an optional extension) may be specified to the right of a "#".

EXAMPLES

These examples show the use of the "#" in an F.I.

Assume "MYLIBRY" is the current library.

Example 1:

```
"#"
```

This example locates all files under "MYLIBRY" as shown in Figure 4-7.

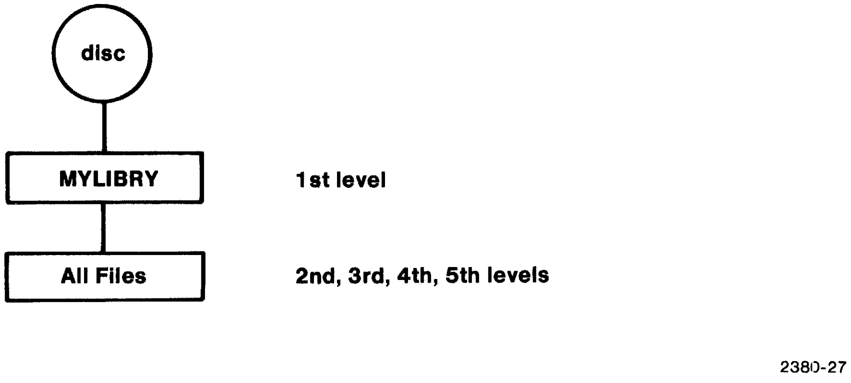


Figure 4-7. Sample Storage Structure (2 Levels).

Example 2:

"A#"

This example locates all files on the 3rd, 4th and 5th levels in library "A" as shown in Figure 4-8.

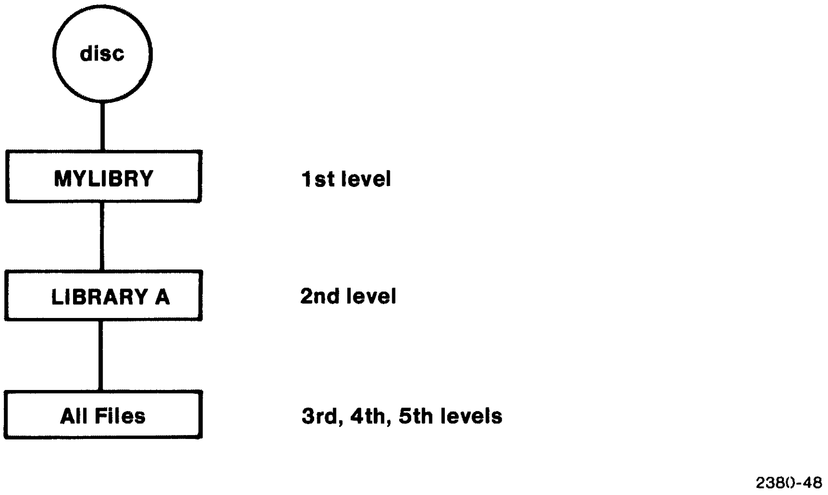
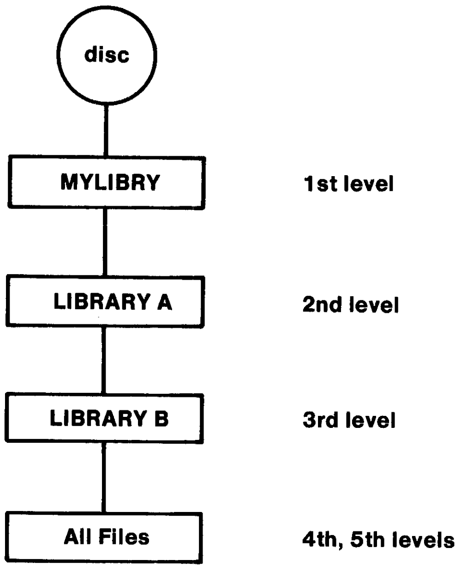


Figure 4-8. Sample Storage Structure (3 Levels).

Example 3:

"A/B#"

This example locates all files on the 4th and 5th levels below library "B" as shown in Figure 4-9.



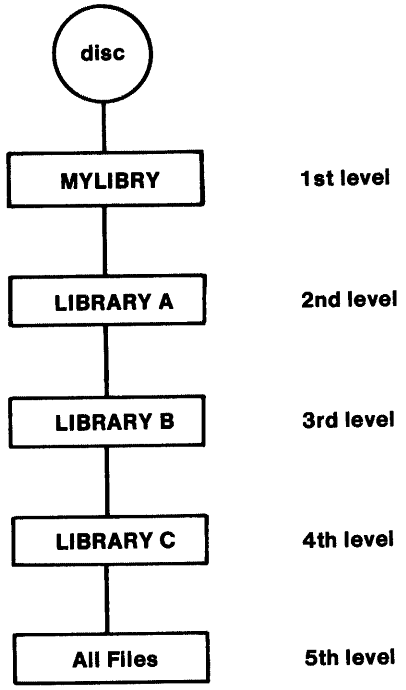
2380-29

Figure 4-9. Sample Storage Structure (4 Levels).

Example 4:

"A/B/C#"

This example locates all files on the 5th level below libraries "A", "B", and "C" as shown in Figure 4-10.



2380-30

Figure 4-10. Sample Storage Structure (5 Levels).

Example 5:

'A/B#.BLUE'

This example locates all files under library "B" with the extension "BLUE". All other files in the storage structure with the extension "BLUE" will be ignored.

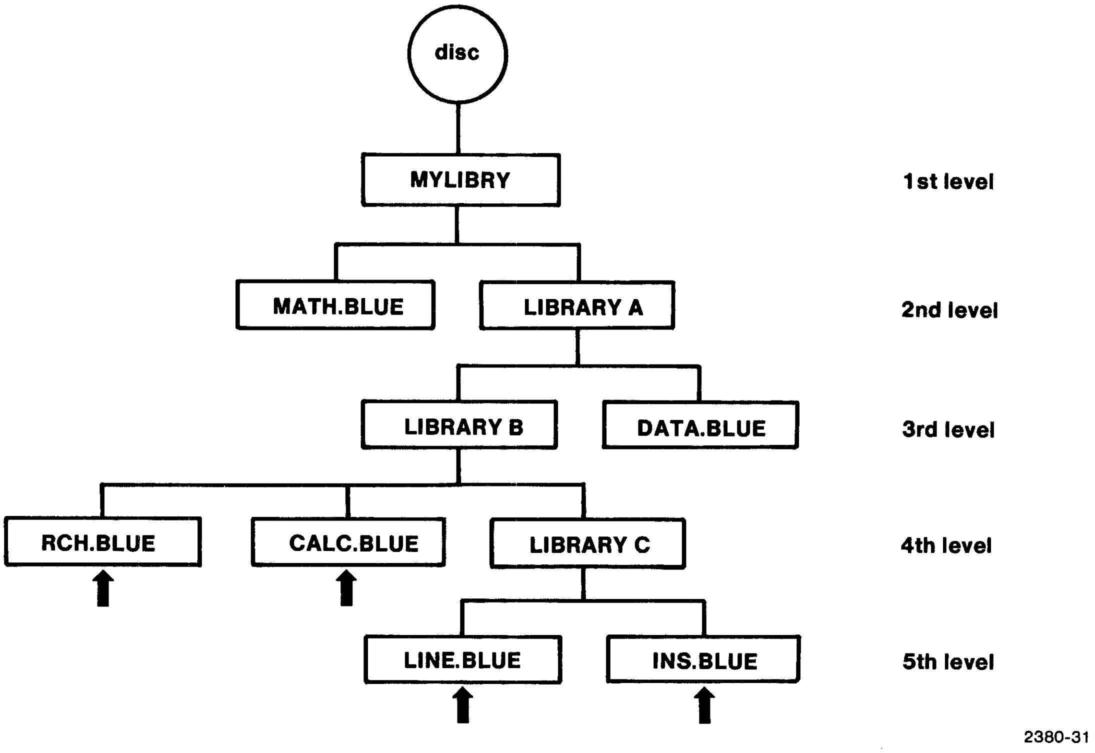


Figure 4-11. Sample Storage Structure (5 Levels).

Asterisk(*)

The asterisk may be used three ways:

1. To select all eligible libraries or files at the level or levels the character is entered.

2. To select all eligible libraries or files with a particular prefix.

3. To select all files using extensions with a particular prefix.

Limitations:

- An asterisk locates only those libraries or files within a single level. All names on preceding and subsequent levels must be specified. If no subsequent names are specified, only files at the asterisk level will be selected.

- An asterisk cannot be used to locate a file or library with a password.

- Characters in a name cannot be used to the right of an asterisk; for example, "DO*" is legal but "D*N" is not.

Examples 1 through 4 show how to locate all eligible libraries or files at the level the character is entered. Examples 5 through 7 show how to locate a file, library or extension with a particular prefix. Examples 5 through 7 are not whole F.I.'s but typical entries usable at any level. Assume that MYLIBRY is the current library for all these examples.

Example 1:

"*/B/C"

This example locates all files named "C" in any 3rd level library named "B" which is in any 2nd level library in "MYLIBRY".

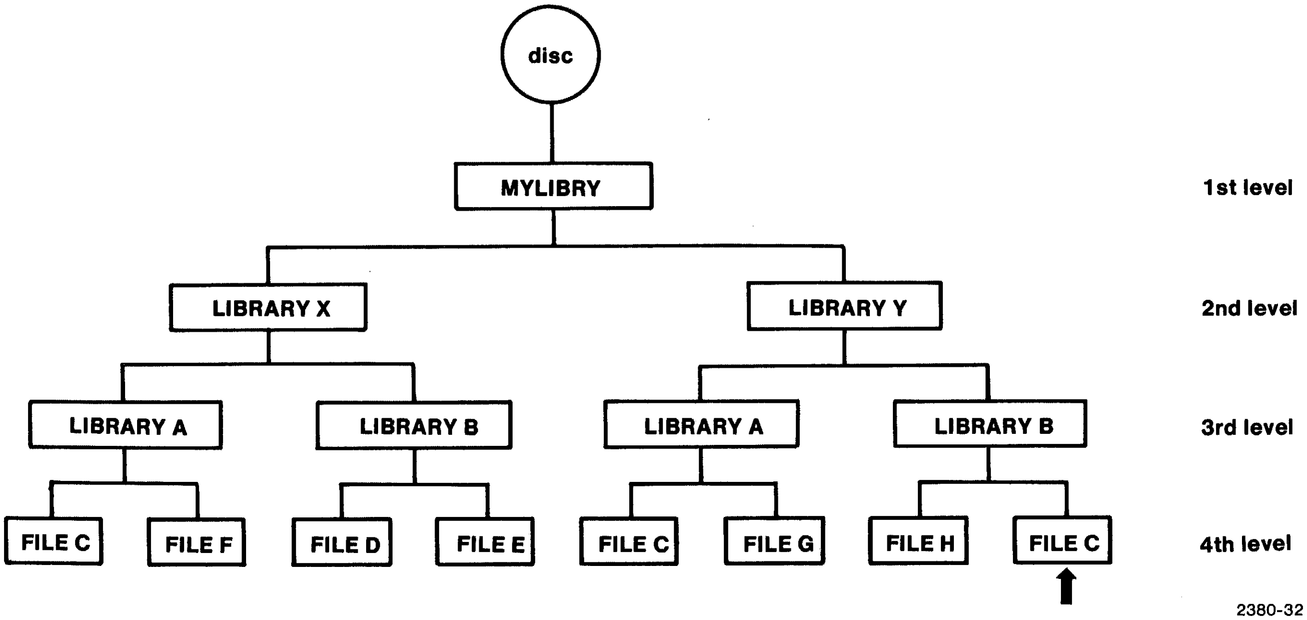


Figure 4-12. Sample Storage Structure (4 Levels, 8 Files).

Example 2:

"A/*/C"

This example locates any file named "C" in any 3rd level library which is in 2nd level library "A" in "MYLIBRY" .

Example 3:

"A/B/*"

This example locates all 4th level files in the 3rd level library "B" which is in the 2nd level library "A."

Example 4:

"A/B.*"

This example locates all 3rd level files "B" with any extension in 2nd level library "A." The arrows in Figure 4-13 show these files.

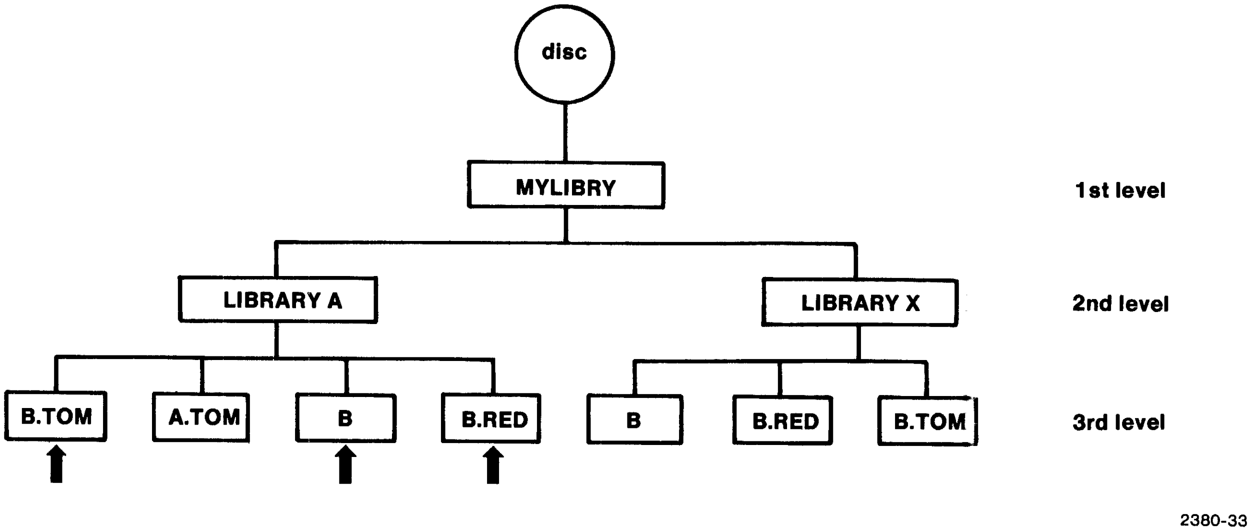


Figure 4-13. Sample Storage Structure (3 Levels, 7 Files).

Example 5:

A*

This example locates any library or a single file with the first letter of "A" such as these:

AB
A1
AABCO

Example 6:

```
A*.RED
```

This example locates any file with the first letter "A" and the extension "RED". Remember that extensions cannot be assigned to library names.

Example 7:

```
JUMP*
```

This example locates any library or file with the prefix "JUMP" such as these:

```
JUMPING
JUMP
JUMPY
```

Question Mark (?)

The question mark may be used to replace single characters when specifying library names, file names or extensions.

Limitation:

The question mark cannot represent more than a single character.

The following examples are not complete F.I.'s but typical entries using the "?" which may be employed at any level. SCRATCHLIB is the current library.

Example 1:

```
?
```

This example locates any library or file with a name one character in length.

Example 2:

```
????
```

This example locates any library or file name one, two, three or four characters in length.

Example 3:

```
???/???
```

This example locates all library names on one level which are one, two, or three characters long with any file on the next level with a name one, two, or three characters long.

Example 4:

```
CA?/CO?
```

This example locates all library or file names on one level with two or three characters beginning with "CA" with any file name on the next level with two or three characters beginning with 'CO.'

The example will locate:

```
CA/CO
CA/COW
CAT/CO
CAB/COT
CAT/COW
CAR/CON
CAP/COB
```

This example will not locate:

```
CATT/COWE
CATT/COW
CAT/COWE
CAB/KING
CA/COWE
CATT/CO
C/C
C/COW
CAT/C
```

USERLIB String Suppression (@)

When a string representing the first field of an F.I. is entered with a CALL "USERLIB" command, that portion is accessed and automatically placed in front of any F.I. in subsequent commands. This portion is called the current library, and its purpose is to reduce the amount of work and space in accessing files in subsequent commands.

However, it is not always desirable to access a file in the current library. To circumvent the current library, a commercial "at" sign (@) must precede the F.I. By using the "@," the system will access any other library named in the F.I.

EXAMPLES:

```
100 CALL "USERLIB","YOURLIBRY"

250 OPEN "@MYLIBRY/A/B";3,"F",A$

300 OPEN "L/M";2,"F",B$
```

When line 100 is executed, "YOURLIBRY" is specified as the current library.

When line 250 is executed, the system accesses "MYLIBRY" because of the "@" in the F.I. and ignores the current library, "YOURLIBRY". This does not change the current library.

When line 300 is executed, the system automatically accesses the current library, "YOURLIBRY," and locates 2nd level library "L" and 3rd level file "M."

Remember:

- The "@" must be used to access a user library UNLESS that library is the current library.
- The current library defaults to SCRATCHLIB.
- The "@" also can be used to create or access a 1st level file in the same way it creates or accesses a library.

Simplified SYSLIB Access ($)

There are three ways to access SYSLIB (system library):

1. Enter SYSLIB into a CALL "USERLIB" command. This makes "SYSLIB" the current library. The $ should not be used in a CALL "USERLIB" command.

2. Enter an "@" as the first character in an F.I. followed by "SYSLIB" and the balance of the F.I.

3. Enter a "$" (which reads "SYSLIB" to the Graphic System as the first character in an F.I. followed by the balance of the F.I. No slash mark (/) is necessary after entering the $.

When method 2 or 3 is employed, the current library is ignored.

SECTION 5

CONTENTS

Section 5

COMMAND DESCRIPTIONS

INTRODUCTION

This section explains how to read command descriptions. Each command is then discussed in detail.

HOW TO READ COMMAND DESCRIPTIONS

The File Manager commands are arranged alphabetically (the word 'CALL' is disregarded).

FORMAT

The individual command descriptions are divided into the following parts:

Purpose
Syntax Form
Descriptive Form
Field Definitions
General Information
Prerequisites
Examples

Occasionally, extra information will be added to a command description to help a user with its execution; for example, FILE OR FILE RECORD SIZE REQUIREMENTS is included in the CREATE command description.

Purpose

This section explains the function of the command.

Syntax Form

The syntax form shows the abbreviated form of the keyword, and the number of fields, delimiters and punctuation required. Remember, quotation marks are required for all strings and string constants. See the examples in the command descriptions.

The syntax form also shows either the exact entry or class of entry required (constant, string constant, numeric expression, numeric variable, string, string variable, target string variable, target numeric variable). Although a file identifier may be represented as a string or string variable, it is always shown in the syntax form as F.I.. For more details see HOW TO WRITE A COMMAND (Section 4).

Brackets [ ] indicate that the field is optional. Nested brackets [A[B]] indicate that while both fields are optional, the outside field (A) has priority and any entry must be made there first.

Braces { } indicate that the field is required. You must choose between the terms, characters, classes or numbers within the braces.

Descriptive Form

This form shows the complete keyword, followed by a one or two-word definition of each field.

Field Definitions

This section gives a complete description of each of the fields shown in DESCRIPTIVE FORM.

General Information

Operational details of the command are described in this section.

Prerequisites

This section lists operations or commands that must be executed before executing the command being discussed. It may also describe the condition that must exist before execution of the command.

Examples

This section shows examples of the command. In most cases a sample line number precedes the command. Some examples represent field contents with X's or N's. The use of commands in programs is demonstrated in INPUT and READ command descriptions as well as SAMPLE PROGRAMS, USING PRINT AND INPUT IN A PROGRAM, and USING WRITE AND READ IN A PROGRAM in the appendices.

NOTE

Line numbers, though not always shown, can precede all commands described in this manual.

APPEND

Purpose

The APPEND command brings a program from a specified file on the disc and places it:

a. Prior to the program already in memory or
b. Within the body of the program already in memory or
c. Subsequent to a program already in memory.

SYNTAX FORM

APP F.I. [, string] ; constant, constant

DESCRIPTIVE FORM

APPEND F.I. [,"ASCII"] ; target line number, increment between line numbers

Command Field Definitions (Descriptive Form)

APPEND

This is the keyword for this statement. Only three characters, APP, are required.

F.I.

This entry must match the F.I. already assigned this file or indicate the location of the F.I. with a string variable. If the F.I. is entered, it must be in quotes: for example, "@MYLIBRY/DOG".

"ASCII"

"ASCII" or a string variable may be entered here. An "ASCII" entry implies that ASCII data is to be retrieved. If a string variable is used, it may represent " ", implying binary data, or "ASCII", implying ASCII data. This way a A$ entry, for example, allows the same command to be used repeatedly to retrieve both kinds of data.

If "ASCII" is entered, it must be in quotes: " ". If no entry is made, the system defaults to a binary append.

target line number — This entry provides the line number of the current program to which the file program will be sent. The entry must be an integer of 1 to 65535.

increment between line numbers — This entry specifies the increment between line numbers. The entry must be an integer of 1 to 65535. If no entry is made in this field, the increment defaults to 10. See Graphic System Reference Manual for details.

General Information

The appended program is inserted into the current program at the target line number. The first statement transmitted from the file program REPLACES the statement already appearing on the target line number. Line numbers in the current program greater than the target line number will be placed after the appended program. Any line numbers following the target line number will be renumbered.

This command is executed on a file in the current device. See UNIT command description.

Prerequisites

Disc must be mounted but device must not be reserved. File does not have to be OPEN.

Examples

Example 1:

```
100 APPEND "@MATH.OBJ";210,5
```

This example appends a program with an F.I. of "MATH.OBJ" to the current program in memory starting with line number 210. The combined program is renumbered in increments of 5 from line 210 up.

Example 2:

```
120 APPEND "@ARCH:J1500.OBJ",110,5
```

This example appends a program with an F.I. of "ARCH:J1500.OBJ." The target line number is 110 and the increment is 5.

Example 3:

```
135 APPEND "$FUNNY.PGM";250
```

This example appends the program in "FUNNY.PGM" located in SYSLIB. The target line number is 250.

Example 4:

```
140 APPEND B$;3500
```

This example appends any program that has its F.I. located in B$. The target line number is 3500.

ASSIGN

Purpose

The ASSIGN command lets you make changes to the file attributes that were originally assigned in the CREATE command.

SYNTAX FORM

ASS "F.I."; string

DESCRIPTIVE FORM

ASSIGN "F.I."; attributes

Field Definitions (Descriptive Form)

ASSIGN — This is the keyword for the command. The entry may be made as shown in syntax form.

F.I. — The entries in this field must match the F.I. currently assigned to this file. An F.I. must be in quotes.

attributes — Except for the BINARY or ASCII designations, any current file attributes may be changed by entering one or more of the following characters in this field.

R(private)
U(public)
S(scattered)
C(contiguous)
N(not compressible)
M(compressible)

S can be changed to C only if the data entered so far is contiguous.
Conflicting entries such as R and U will result in the last, or "right-most", character being accepted. See CREATE command description.
An attribute must be in quotes: "R".

General Information

Generally, this command is executed from the keyboard rather than as part of a program.

This command is always executed on a file in the current device. See UNIT command description.

Prerequisites

Device must be mounted but not reserved.

Examples

Example 1:

```
ASSIGN"@FINANCE/CNTRYBNK.OBJ";"R"
```

This command changes the "CNTRYBNK.OBJ" file in "FINANCE" to a private status.

Example 2:

```
ASSIGN"$MATH.OBJ";"RSN"
```

This command changes the "MATH.OBJ" file in the SYSLIB to a private, scattered and non-compressible status.

Example 3:

```
ASSIGN"$MATH.OBJ";A$
```

This command carries out the same function as example 2 if A$ = "RSN."

CLOSE

Purpose

The CLOSE command is used to close all open files or to close a specific file.

SYNTAX FORM

```
CLO [constant]
```

DESCRIPTIVE FORM

```
CLOSE [lfn]
```

Field Definitions (Descriptive Form)

CLOSE — This is the keyword for this command. Only three characters, or CLO, are required.

lfn — Logical file number: The entry in this field specifies the particular file to be closed. This number, always an integer from 1 to 9, must match the logical file number assigned in the OPEN statement. When no number is assigned, all files in the Graphic System internal tape unit and the external disc devices are closed.

General Information

See FILE POINTER OPERATION (Appendix C), which shows where the logical pointer is located after a file is closed. This is significant when the file is reopened for another read, write or update operation.

This command is always executed on files in the current device. See UNIT command description. If the file specified in the command is not open, the CLOSE command is ignored.

OLD, END, INIT or DELETE ALL commands close all open files. Pressing the BREAK key twice also closes all open files but aborts program control of files.

Prerequisites

None.

Examples

Example 1:

```
1100 CLOSE 7
```

This example closes a file that has been assigned logical file number 7.

Example 2:

```
1005 CLOSE
```

This example closes all files on both the internal tape drive and all external disc devices.

CALL "COMPRS" (Compress)

Purpose

The CALL "COMPRS" command collects unused spaces on a disc into one contiguous space. You have the option of reducing existing file space on each file to exactly fit stored data or programs. This makes more space available for new files.

SYNTAX FORM

CAL "COMPRS", numeric expression, numeric expression

DESCRIPTIVE FORM

CALL "COMPRS", device address, compress control

Field Definitions (Descriptive Form)

CALL "COMPRS"	These are the keywords for this command. Entry may be made as shown in SYNTAX FORM.
device address	The device address may be seen on or near the front of the device.
compress control	When 1 is entered, the system collects and groups all space not already occupied by data or programs regardless of whether or not that space is formatted as file space. When 0 is entered, the system collects and groups only that space not occupied by files.

General Information

While CALL "COMPRS" is quite useful, it cannot collect all the free space on the disc. It also may cause the system to run slower on files that have been expanded after being compressed. To collect and group 100% of the free space, a CALL "DUP" command must be executed.

An unmarked bad block may be encountered during a COMPRESS execution. This results in a device I/O error. There is no way to tell which file contains the bad block unless each file is read. When the bad block in the file is encountered, an error message containing the bad block address is displayed. This address must be entered in a CALL "MRKBBG" command. After this command has been executed, CALL "COMPRS" may be issued.

If the bad block is contained in the directory, a CALL "DUP" command is necessary for information recovery. The duplicated disc information will be missing one or more library structures corresponding to the extent of damage or wear in the directory. If the available system contains only a single drive, the disc cannot be read and must be discarded.

Prerequisite

Device must be reserved.

Examples

Example 1:

```
110 CALL "COMPRS",B2,A1
```

The address of the device is in B2. First, the system checks the address; then it finds the device. If 1 has been previously entered in A1, this example collects and groups all space inside and outside files.

Example 2:

```
140 CALL "COMPRS",2,0
```

This example, with actual values in the last two fields, locates the device with the address of 2 and then collects and groups all space not occupied by files (as specified by 0).

COPY . . . TO

Purpose

The COPY . . . TO command will copy one or more files named in an F.I. from one device to another or to a different location on the same disc. All attributes, as well as passwords and extensions, are also duplicated. Changes to names of files, passwords and extensions may also be made using this command.

SYNTAX FORM

COP "F.I.", numeric expression TO "F.I.", numeric expression

DESCRIPTIVE FORM

COPY "F.I." (source), source device address TO "F.I." (target), target device address

Field Definitions (Descriptive Form)

COPY . . . TO

These are the keywords for this command. They may be entered as shown in SYNTAX FORM.

F.I. (source)

The F.I. source entry is the F.I. of the file to be copied. If more than one file is to be duplicated or if F.I. changes are desired, see MULTIPLE FILE TRANSFERS or CHANGING NAMES in this command description.

An F.I. must be in quotes.

source device address

This is the address of the device that contains the files. This address is shown on or near the front of the device.

F.I. (target)

The F.I. target entry defines where the file is to be copied and the name assigned to the copied file. Remember, two files with the same F.I. cannot be duplicated on the same disc.

An F.I. must be in quotes.

target device address

This is the address of the device that is to receive the information. The address is shown on or near the front of the device.

General Information

Duplication will not occur if:

- F.I. target already exists on the second device.
- The source F.I. is a library.
- The source device or target device is reserved or write-protected.
- The disc is not mounted or is incorrectly mounted.
- The second device contains bad blocks.
- Specified passwords are not used.
- More than 8 files are open.

This command is also used to copy files from one area to another on the same disc.

Libraries are created in the new location if none exist.

No files are erased.

Prerequisites

Discs must be mounted but the device must not be reserved. Files must be closed.

Examples

Example 1:

```
110 COPY“$DOG/CAT”,2TO“@MYLIBRY/DOG/CAT”,1
```

Assume SCRATCHLIB is the current library.

This example accesses SYSLIB on device 2 and copies file "CAT" in library "DOG" to MYLIBRY in device 1.

Example 2:

```
125 COPYA$,ATOB$,B
```

This example functions exactly as Example 1, if the F.I. and device address have previously been entered into the variables shown:

```
A$ = "$DOG/CAT"

A = 2

B$ = "@MYLIBRY/DOG/CAT"

B = 1
```

Example 3:

```
450 COPY"FRED/TOM",1TO"@YOURLIB/FRED/TOM",2
```

This example presumes that a CALL "USERLIB" has been executed specifying the 1st level library in the source F.I..

Library "FRED" and file "TOM" in device 1 are copied to YOURLIB in device 2.

Example 4:

```
150 COPY"FRED/TOM",1TO"@YOURLIB#",2
```

This example functions like Example 3. The only difference is in the construction of the target F.I.. A pound sign (#) has been used to indicate that everything being transmitted, including the F.I., should be duplicated as it is. No slash (/) is necessary between the 1st level library (YOURLIB) and the "#."

Multiple File Transfers. Using "#", "*" or the "?" (see SPECIAL CHARACTERS in Section 4) in the F.I. construction allows selection and duplication of specific groups of files. The following examples illustrate how these characters may be used. Those libraries and files with passwords are ignored unless the password is part of the F.I.

Assume "MYLIBRY" is the current library.

The Pound Sign (#)

Example 1:

```
120 COPY "#",1TO"#",2
```

This example duplicates everything in "MYLIBRY" on device 1 to "MYLIBRY" on device 2. This statement would not be carried out if both device addresses were identical.

Example 2:

```
180 COPY"DOG#",1TO"#",2
```

This example duplicates everything in the 2nd level library "DOG" in "MYLIBRY" on device 1 to "MYLIBRY" on device 2.

Example 3:

```
460 COPY"@YOURLIB#",1TO"#",0
```

This example duplicates everything in YOURLIB on device 1 to "MYLIB" on device 0.

Example 4:

```
180 COPY"#.CAT",1TO"#",2
```

This example duplicates all files with the extension "CAT" contained in "MYLIBRY" on device 1 to "MYLIBRY" on device 2.

The Asterisk (*)

Example 1:

```
180 COPY"MATH/*/ALG",1TO"#",2
```

This example selects any file named "ALG" in any library in a library named "MATH." These files are located in "MYLIBRY" on device 1 and are duplicated in "MYLIBRY" on device 2.

Example 2:

```
170 COPY“*/DOG/CAT/MOUSE”,1TO“#”,2
```

This example selects all 2nd level libraries with subsequent libraries of “DOG” and “CAT” with files of “MOUSE.” These are located in “MYLIBRY” on device 1 and are duplicated on MYLIBRY on device 2.

Example 3:

```
285 COPY“FAST*/*/*”,1TO“#”,2
```

This example selects all 2nd level libraries with the prefix “FAST” and subsequent 3rd and 4th level libraries and files. If a library with this prefix only contains files to the 3rd level or files that continue to the 5th level, no files will be selected.

These libraries and files will be selected:

FAST/RED/WHITE.YELL
FASTLINE/RED/GREEN
FASTLOOSE/YELLOW/BLUE

These libraries and files will not be selected:

FAST/RED/WHITE/YELLOW
FAST/RED/WHITE:PASS
GO/JIM/GREEN
GOSOON/LIGHT/GOOD

Files selected are duplicated from “MYLIBRY” on device 1 to “MYLIBRY” on device 2.

Question Mark (?)

Example 1:

```
170 COPY“GO?/TOM/FRED”TO“#”2
```

This example selects all 2nd level libraries with a two or three character prefix beginning with “GO”. These libraries will be selected only if they contain a subsequent library of “TOM” and a file of “FRED”.

These libraries and files will be selected:

GO/TOM/FRED
GOT/TOM/FRED
GOB/TOM/FRED
GOG/TOM/FRED.OBJ

These libraries and files will not be selected:

GOT/TOM/FRED/BOB
GOD/TOM/FRED:PASS
GOTT/TOM/FRED

Libraries and files selected are duplicated from "MYLIBRY" on the current device to "MYLIBRY" on device 2.

Example 2:

```
210 COPY"???/*/*",1TO"@YOURLIB#",2
```

This example selects all libraries with up to three character names with libraries in the 3rd level and files in the 4th level. They are duplicated to "YOURLIB" on device 2.

These files will be selected:

TAD/OD/RATES
D33/ZIP/NONE.OBJ
DO/CONS/PROJ

These files will not be selected:

TAD/ALG/MATH/COMP
GILD/TOM/FRED

Changing Names. Name changing may be accomplished while duplicating single files, without the use of special characters. Examples 1 through 3 show how to change names when single files are duplicated. Name changing while duplicating multiple files, however, requires using "#", "*", or the "?" (described in SPECIAL CHARACTERS IN FILE IDENTIFIERS in Section 4). These allow the user to change the names of libraries, files, passwords and extensions when transferring multiple files. Example 4 illustrates how these characters may be used. Assume "MYLIBRY" is the 1st level or current library.

Example 1:

```
120 COPY"@YOURLIB/TOM",1TO"BOB",2
```

This example selects the file "TOM" in "YOURLIB" on device 1. The file is duplicated as "BOB" on device 2 in "MYLIBRY." Remember, if "TOM" is a library, no duplication will take place.

Example 2:

```
130 COPY"@YOURLIB/KEN/FRED:BEN",1TO"JANE/LORI:PAM",2
```

This example selects the file "FRED" with the password "BEN" in library "KEN" in "YOURLIB" on device 1. The file is duplicated on device 2 in "MYLIBRY" as file "LORI" with the password "PAM" under the new library "JANE."

Example 3:

```
275 COPY"A/B/C",3TO"$M/N/O",2
```

This example selects the file "C" in libraries "MYLIBRY", "A", and "B" on device 3. The libraries and file are duplicated on device 2 in SYSLIB as "M", "N", and "O."

Example 4:

```
270 COPY"$A#",1TO"B#",2
```

This example selects all files in library "A" in "SYSLIB" on device 1. These files are duplicated to "MYLIBRY" on device 2. The name of library "A" is changed to "B."

CREATE

Purpose

The CREATE command creates file space on the disc, and assigns the attributes and F.I., including passwords and extensions, if any. The last two fields in this command specify whether the file is to be random or sequential.

SYNTAX FORM

CRE "F.I." [,"string"] ; numeric expression, numeric expression

DESCRIPTIVE FORM

CREATE "F.I." [,"attributes"] ; number of logical records, record length

Field Definitions (Descriptive Form)

CREATE	This is the keyword for this statement. Only three letters, CRE, are required.
F.I.	The entries in this field must meet F.I. requirements. See HOW TO WRITE A FILE IDENTIFIER in Section 4. An F.I. must be in quotes.
attributes	If no entry is made here, the system defaults to the following conditions: B (BINARY): Only binary data may be entered. R (PRIVATE): Read operations may not be executed without using the password if a password has been specified. S (SCATTERED): Files will be located wherever there is space (on a single disc only).

N (NOT COMPRESSIBLE): Files may not be compressed when file space exceeds current stored program or data.

Any or all of the above conditions may be changed by entering one or more of the following attributes:

A (ASCII): Only ASCII data may be entered.

U (PUBLIC): Read operations may be executed without using a password in a OPEN "G" command.

C (CONTIGUOUS): File can be stored only as a whole entity in a single location.

M (COMPRESSIBLE): File may be compressed if in excess of current stored program or data.

H (HOST BINARY): File can only be used to store binary programs.

Any attribute entry must be in quotes: "M".

number of logical records

This entry specifies the number of records to be located within a random file, or, if the record length is zero, the length in bytes of a sequential file. To see how large your file must be, see FILE OR FILE RECORD SIZE REQUIREMENTS later in this description.

record length

If file is to be random:

This entry specifies the length of the records in bytes. These records are generally specified just long enough to contain the items or strings to be stored. See FILE OR FILE RECORD SIZE REQUIREMENTS at the end of this description.

If file is to be sequential:

Enter zero (0). The number entered in the previous field is then the length of the file.

General Information

The system automatically extends a file during WRITE, PRINT or SAVE operations if information exceeds allocated space. If the file has been designated C (contiguous) and there is an immediately adjacent file, an error message appears. Files cannot be extended over adjacent files in this way. If an error message appears, execute an ASSIGN command to change the attribute "C" to "S."

This command is executed on a file in the current device. See UNIT command descriptions.

File space is always extended in multiples of 256 bytes.

Prerequisites

Disc must be mounted but device not reserved. File must not already exist.

Examples

Example 1:

```
100 CREATE "$MATH:RED.OBJ","U";100,128
```

This command specifies that the file with the F.I. shown ($MATH:RED.OBJ) is to be public (U) which means it may be read without use of the password "RED." It also specifies there are to be 100 records of 128 bytes each. The 128 indicates this is a random file. If the last entry were zero (0), the file would be a sequential file of 100 bytes.

Example 2:

```
100 CREATE A$,"UA";100,128
```

This statement specifies that the entire F.I. is stored in A$ and that the file is to be public (U) and ASCII (A). This means it may be input without use of the password (if any). It also specifies there are to be 100 records of 128 bytes each.

Example 3:

```
110 CREATE B$,C$;A,0
```

This statement specifies that the F.I. is stored in B$ and that the attributes will be found in C$. The number of logical records normally is specified by the variable A, but because zero was placed in the record length file, the value of A will designate file length instead of number of records. This means this will be a sequential file.

Example 4:

```
110 CREATE B$,C$;A,B
```

This statement specifies that the F.I. is stored in B$ and that the attributes will be found in C$. The numbers and length of logical records in the file is specified by the values of A and B, respectively.

Example 5:

```
140 CREATE"@MYLIBRY/JACK";1000,0
```

This statement creates a sequential file named "JACK" in "MYLIBRY" with default attributes and 1000 bytes in length.

File or File Record Size Requirements

The number of bytes allocated to a particular file or file record depends on whether you will be writing or printing information to that file.

The table below shows how many bytes are required in PRINT and WRITE.

ASCII (PRINT)

Numeric values and strings	1 byte per character + 1 byte for Carriage Return

BINARY (WRITE)

Numeric Values	9 bytes for any value + 1 byte for EOR*
Strings	4 bytes + 1 byte for each character + 1 byte for EOR*

*EOR = End of Record Item

See the Graphic System Reference Manual for additional information about multiple items in a PRINT statement and the PRINT . . . USING statement.

Remember that the "dynamic allocation" feature extends file space as information is entered. Both sequential and random files are extended in blocks of 256 bytes. Individual records in random files, however, cannot be extended in size. For this reason, it is important that you know the size requirements of the data you will be entering in random files *before* you create those files.

CALL "CUSTAT" (Controller Unit Status)

Purpose

The CALL "CUSTAT" command generates status messages for all devices (disc drives) interfaced to the 4907 File Manager controller. This command does the same thing CALL "DSTAT" does except it generates messages for all devices, rather than a specified device.

SYNTAX FORM

CAL "CUSTAT", string variable

DESCRIPTIVE FORM

CALL "CUSTAT", target string variable

Field Definitions (Descriptive Form)

CALL "CUSTAT"

These are the keywords for the command. They may be entered as shown in SYNTAX FORM.

target string variable

This is where device status messages are sent. The target string variable must be dimensioned large enough to accommodate the messages. Each device status message is 186 characters.

General Information

If the controller is designed to support more devices than are interfaced, the CALL "CUSTAT" command returns meaningless device status messages.

Prerequisites

Set clock.

Example

```
CALL "CUSTAT", A$
```

This example generates device status messages for all devices and stores them in A$.

DELETE ALL

Purpose

This command erases everything currently in the Graphic System's memory.

This is a 4050 SERIES BASIC command. It is included here because it directly affects 4907 system operation.

SYNTAX FORM

DEL ALL

DESCRIPTIVE FORM

DELETE ALL

General Information

In addition to erasing the Graphic System's memory, this command closes all disc files, sets the current device to zero (0), and specifies "SCRATCHLIB" as the current library.

Prerequisites

None.

DIRECTORY

Purpose

The DIRECTORY command generates file status messages which may be displayed on the Graphic System screen or stored on the Graphic System internal magnetic tape. If the Graphic System is equipped with Option 10, status messages may be output to the printer.

SYNTAX FORM

$$\text{DIR}\left[\left[\begin{Bmatrix}\text{@N:}\\ \text{@33:}\end{Bmatrix}\right]\left[\begin{Bmatrix}0\\1\\2\end{Bmatrix}\left[,\text{"F.I."}\right]\right]\right]$$

DESCRIPTIVE FORM

DIRECTORY [[I/O address] [format code [,"F.I."]]]

Field Definitions (Descriptive Form)

DIRECTORY

This is the keyword for this command. Only the first three characters, DIR, are necessary.

I/O address

If no entry is placed in this field, the system displays the messages on the Graphic System screen. If the optional 33: is entered, the messages are stored on the Graphic System internal magnetic tape. In this event, a file must first be prepared to accept the information with the BASIC FIND and MARK commands. Each full file status message is 189 characters plus the characters in the F.I.
N represents the address of the printer if Option 10 is used.

format code

If there is no entry in this field or the entry is zero (0), the command returns a status message containing only the F.I. (name) of the file or files specified in the last field of this command. The format code field cannot be left blank if an entry is made in the F.I. field.

Entering 1 returns F.I.'s as well as relevant times and dates involving the file or files specified in the last field of this command.

Entering 2 returns messages describing the "full file status." This means that not only are the F.I.'s time and dates returned but also information regarding file space and attributes.

No passwords are returned when executing DIRECTORY. Further information about file status messages may be seen in DEVICE AND FILE STATUS MESSAGES (Appendix A).

F.I.

The system looks in this field for the F.I. of the file or files to be selected for a status check. If no F.I. entry is found, the system assumes that a status check is desired for ALL files in the current library.

Special characters may be used in this field if special or multiple file selection is desired.

Any F.I. must be in quotes.

General Information

This command is executed on a file in the current device. See UNIT command description.

Prerequisites

Disc must be mounted.

Examples

Assume, for Examples 1-3, that SCRATCHLIB is the current library.

Example 1:

```
100 DIR0,"@MYLIBRY#"
```

This example generates status messages for all eligible files in "MYLIBRY." Since there is no I/O address entry, all messages are displayed on the Graphic System screen. The zero entry in the format code specifies that the status messages will contain only the F.I.'s of the selected files.

Example 2:

```
100 FIND 10
110 MARK 1,1000
120 FIND 10
130 DIRECTORY @33:2,"$STAT/TESTDESIGN"
140 CLOSE 0
```

This example locates the file "TESTDESIGN" in the library "STAT" in SYSLIB. The file status is then checked and a "full status message" is stored on the Graphic System internal magnetic tape. The message is stored on file 10 on the tape which previously has been MARKed or dimensioned to 1000 bytes.

Example 3:

```
100 DIR
```

Because there are no entries in the optional fields in this example, default conditions apply. The above command is equivalent to the command:

```
100 DIRECTORY @32:0,"@SCRATCHLIB#"
```

This example displays on the Graphic System screen "name only" file status messages for all files contained in SCRATCHLIB.

Example 4:

Assume for this example that MYLIBRY is the current library.

```
100 DIR
```

Because there are no entries in the optional fields in this example, default conditions apply. The above command actually appears like this to the system:

```
100 DIRECTORY @32:0,"@MYLIBRY#"
```

This example displays on the Graphic System screen "name only" file status messages for all files contained in MYLIBRY.

Example 5:

```
100 DIR0,"@"
```

This example displays on the Graphic System screen only the names of all the files on the current disc.

CALL "DISMOUNT"

Purpose

The CALL "DISMOUNT" command notifies the system that a disc is to be removed from active use. After DISMOUNT execution, no OPEN statements may be executed on the disc. The term "DISMOUNT" does not mean that the disc is physically removed from the drive.

SYNTAX FORM

CAL "DISMOUNT", numeric expression

DESCRIPTIVE FORM

CALL "DISMOUNT", device address

Field Definitions (Descriptive Form)

CALL "DISMOUNT" These are the keywords for this statement. The entry may be made as shown in SYNTAX FORM.

device address The device address may be seen on or near the front of the device.

General Information

All files must be closed before a DISMOUNT command will take effect. If a file is open at the time of execution, the system waits until it is closed. File I/O, however, may continue. After a dismount, no files may be opened on that device until a CALL "MOUNT" command is executed.

Prerequisites

Files must be closed.

Examples

Example 1:

```
115 CALL "DISMOUNT",C1
```

This example tells the system that the device with its address in C1 is to be dismounted.

Example 2:

```
155 CALL "DISMOUNT",2
```

This example informs the system that the device with the address 2 is to be dismounted.

CALL "DREL" (Device Release)

Purpose

When a device is reserved it must subsequently be released in order to open files. The CALL "DREL" command releases the addressed device previously reserved with a CALL "DRES" command.

SYNTAX FORM

CAL "DREL", numeric expression

DESCRIPTIVE FORM

CALL "DREL", device address

Field Definitions (Descriptive Form)

CALL "DREL"	These are the keywords for this command. Entry may be made as shown in SYNTAX FORM.
device address	The device address is shown on or near the front of the device.

General Information

The addressed device must be reserved, or the statement will be ignored.

Prerequisites

None.

Examples

Example 1:

```
110 CALL "DREL",B2
```

This example releases the device that has its address stored in B2.

Example 2:

```
175 CALL "DREL",2
```

This example releases the device with address 2.

CALL "DRES" (Device Reserve)

Purpose

The CALL "DRES" command provides exclusive device control. This command is necessary when formatting a file or when the CALL "COMPRS" or CALL "DUP" commands are executed.

SYNTAX FORM

CAL "DRES", numeric expression

DESCRIPTIVE FORM

CALL "DRES", device address

Field Definitions (Descriptive Form)

CALL "DRES"
These are the keywords for this statement. Entry may be made as shown in SYNTAX FORM.

device address
The device address may be seen on or near the front of the device.

General Information

All files on the device must be closed before command execution. An error message appears if there are any open files. This command may be executed even if the disc is missing or the device is not up to speed or has an open device door.

Prerequisites

All files must be closed and the clock must be set.

Examples

Example 1:

```
105 CALL "DRES",A
```

This example reserves the device with its address stored in A.

Example 2:

```
115 CALL "DRES",3
```

This example reserves the device with address 3.

CALL "DSTAT" (Device Status)

Purpose

The CALL "DSTAT" command determines the status for the device addressed. See DEVICE AND FILE STATUS MESSAGES (Appendix A). CALL "DSTAT" generates and stores the same status message produced by executing a CALL "MOUNT" command. CALL "DSTAT", unlike CALL "MOUNT", also generates "full" file status messages on any open files, in addition to the device status message.

SYNTAX FORM

CAL "DSTAT", numeric expression, string variable

DESCRIPTIVE FORM

CALL "DSTAT", device address, target string variable

Field Definitions (Descriptive Form)

CALL "DSTAT"

These are the keywords for this command. Entry may be made as shown in SYNTAX FORM.

device address

The device address may be seen on or near the front of the device.

target string variable

The file and device status messages are sent to this location. This string variable must first be dimensioned large enough to hold the device status message.

The device status message is 186 characters. Each "full" file status message is 189 characters plus the number of characters in the F.I.s.

General Information

See DEVICE AND FILE STATUS MESSAGES (Appendix A) for message details.

Prerequisites

Set clock.

Example

```
190 DIM A$(3000)
200 CALL "DSTAT",2,A$
210 PRINT A$
```

This example first dimensions A$ to 3000 bytes. It then requests the status of the device at address 2. The message(s) is stored in A$. Execution of line 210 then prints the messages on the Graphic System screen.

CALL "DUP" (Duplicate)

Purpose

The CALL "DUP" command copies all files and libraries from one device to another. This command is often used to group all unused spaces in a disc into a single contiguous space on a different disc. This command may be used to recover data from a defective disc. This command is not applicable to single device systems.

SYNTAX FORM

CAL "DUP", numeric expression, numeric expression, numeric expression

DESCRIPTIVE FORM

CALL "DUP", source device address, target device address, compress control

Field Definitions (Descriptive Form)

CALL "DUP" — These are the keywords for this statement. Entry may be made as shown in SYNTAX FORM.

source device address — The device address is shown on or near the front of the device. This is the device that contains the files to be copied.

target device address — The device address is shown on or near the front of the device. This is the device which will receive the information.

compress control — When 1 is entered, all free space transmitted will be collected and grouped as one contiguous space on the target device. When 0 is entered, all free space transmitted, except that contained within files, will be collected and grouped as one contiguous space on the target device.

General Information

- The disc receiving the information must first be formatted. Any entry in the master password field in the volume label on the new disc is replaced by the master password being transmitted from the old disc. No other volume label information from the output disc is copied to the new disc.

- When the CALL "DUP" command is executed, ALL INFORMATION PREVIOUSLY STORED ON THE TARGET DEVICE IS ERASED.

- All programs transmitted are duplicated as whole entities. No files are duplicated in "scattered" form.

- This command is ignored if more than eight files are open.

- The CALL "DUP" command does not transfer information to the internal magnetic tape or any outside device.

Prerequisites

A CALL "DRES" command must be executed on both devices.

Examples

Example 1:

```
110 CALL "DUP",D1,D2,A1
```

This example transmits all file information from the device with its address in D1 to the device with its address in D2. If A1 contains the value of 1, then all space, whether or not formatted as file space, will be duplicated as a single contiguous entity on the target device.

Example 2:

```
215 CALL "DUP",1,2,0
```

All file information is transmitted from device 1 to device 2. Since the last field is 0, only that space outside file boundaries is collected and grouped.

END

Purpose

END, a 4050 Series BASIC command, stops all program execution, closes all files and returns control to the Graphic System's keyboard.

SYNTAX FORM

END

DESCRIPTIVE FORM

END

General Information

Executing the END command does not affect the current device or the current library.

Prerequisites

None.

CALL "FFRMT" (Fast Format)

Purpose

The CALL "FFRMT" command operates like CALL "FORMAT". The only difference is that this command does not perform a surface analysis. Instead, the bad block or sector information from the existing volume label is placed on the new volume label created with this CALL "FFRMT" command. As a result, the formatting is faster.

Syntax, Descriptive Forms, and Field Definitions

These are identical to those used in the CALL "FORMAT" command, with the keywords CALL "FFRMT" replacing CALL "FORMAT."

General Information

Attempting a CALL "FFRMT" on a disc not previously formatted or a disc with a damaged volume label results in an error message.

Unless full formatting (CALL "FORMAT") is carried out, bad block or sector information carried over from the previous formatting with a CALL "FFRMT" command may prevent full use of the disc.

This command automatically executes CALL "MOUNT" after its execution, but no device status message is generated.

Prerequisites

Device must be reserved.

CALL "FILE"

Purpose

The CALL "FILE" command generates a file status message for the specified file and stores it in a string variable. Status messages for groups of files may be generated also.

SYNTAX FORM

CAL"FILE", numeric expression, "F.I.", string variable

DESCRIPTIVE FORM

CALL "FILE", device address, "F.I.", target string variable

Field Definitions (Descriptive Form)

CALL "FILE"

These are the keywords for this command. Entry may be made as shown in SYNTAX FORM.

device address

This is the address of the device containing the disc. The address is shown on or near the front of the device.

F.I.

This is the F.I. for the file to be accessed. Status messages from multiple files may be generated with use of special characters in the F.I. See SPECIAL CHARACTERS in Section 4. Passwords specified in the CREATE command are not necessary when writing the F.I. for this command.

An F.I. must be in quotes.

target string variable

This is the location to which the message or messages will be sent. This variable must first be dimensioned large enough to accommodate the incoming messages. Each full file status message is 189 characters long plus the characters in the F.I.'s.

If the file does not exist, the target string receives a null string.

General Information

All messages generated are "full file status" messages and contain pertinent times and dates, file space details and the complete F.I. See DEVICE AND FILE STATUS MESSAGES (Appendix A) for details on the file status messages.

A CALL "FILE" command may be executed even when the device has been reserved. The command may also be executed without a prior OPEN command.

This command will be ignored if more than eight files are open.

Prerequisites

Disc must be mounted.

Examples

Assume SCRATCHLIB is the current library.

Example 1:

```
100 CALL "FILE",2,"@MYLIBRY#",A$
```

This example accesses device 2 and generates status messages for all files in MYLIBRY. The messages are then stored in A$.

Example 2:

```
100 CALL "FILE",A,A$,B$
```

This example accesses the device specified in A and generates status messages for the file or files specified in A$. The messages are then stored in B$.

CALL "FMVALS" (File Manager Values)

Purpose

The CALL "FMVALS" command sends the number of the current device to the numeric variable specified and the name of the current library to the string variable specified. These values can be displayed by recalling the variables.

SYNTAX FORM

CAL "FMVALS", numeric variable, string variable

DESCRIPTIVE FORM

CALL "FMVALS", target numeric variable, target string variable

Field Definitions (Descriptive Form)

CALL "FMVALS" — These are the keywords for the command. They may be entered as shown in SYNTAX FORM.

target numeric variable — This is where the address of the current device is stored.

target string variable — This is where the name of the current library is stored.

General Information

The current device is specified by the UNIT command. The current library is specified by the CALL "USERLIB" command. The defaults are 0 for the device and SCRATCHLIB for the current library.

Prerequisites

None.

Example

```
100 CALL "FMVALS",A,A$
```

CALL "FORMAT"

Purpose

The CALL "FORMAT" command prepares a new disc for data storage. The command will:

- Record a volume "label" on the disc.The volume label contains disc identification and specification data entered in this command.
- Look for bad block or track locations and then place this information on the label. The information on the label may be reviewed by executing CALL "DSTAT" or CALL "MOUNT".

SYNTAX FORM

CAL"FORMAT", numeric expression, "string", numeric expression, numeric expression, "string", "string", numeric expression, numeric expression, numeric expression, numeric expression, numeric expression.

DESCRIPTIVE FORM

CALL"FORMAT", device address, "volume identification", volume number, number of volumes, "owner I.D.", "master password", 1st level chains, 2nd level chains, 3rd level chains, 4th level chains, 5th level chains.

Field Definitions (Descriptive Form)

CALL "FORMAT"
These are the keywords for this command. The entry can be made as shown in SYNTAX FORM.

device address
The device address is shown on or near the front of the device.

volume identification
This entry is the "name" of the disc. The entry may be 1 to 10 characters. The first character must be alphabetic; the rest may be alphanumeric. The volume I.D. must be in quotes: "TOM".

volume number

Enter 1.

number of volumes in series

Enter 1.

These values are not implemented for the 4907 File Manager.

owner identification

This entry can indicate the name of the person formatting the disc, the department with files on the disc, the primary user, etc. The entry can be up to 24 characters, which may be any combination of letters and numbers. The owner I.D. must be in quotes: "FRED".

put name of disk here
(ie "GRAPHICS - BACKUP)
(I.E. "WORK [illegible]

master password

put "PASS"

The master password can be used in CALL "MRKBBG" and KILL. This affects the files in those commands, even if the specified passwords for those files are not used.

The password can be up to 10 characters. The first character must be alphabetic; the rest must be alphanumeric. The master password must be in quotes: "RED".

1st level chains
2nd level chains
3rd level chains
4th level chains
5th level chains

The entry in each of these 5 subfields specifies the number of chains leading to consecutive storage levels. For example, an entry of 2 in the first subfield specifies that two chains lead to the first level.

Each chain can be thought of as a separate storage aid searching through its level for the library or file you've specified. The more searchers you specify, the faster the item you are searching for will be found. With fewer items on a file, more chains bring diminishing returns. More chains require extra room on a disc.

To compute the optimum number of chains between levels, divide the expected number of files or libraries at any level by 5. The result is the approximate number of chains leading to that level (with maximum of 10 and minimum of 1). For example, if you are formatting a disc with

5 first level libraries
25 second level libraries
50 third level files
No fourth or fifth level data

then the entry in this field would be 1, 5, 10, 1, 1.

If your disc will contain no libraries (except first-level default libraries), the level chains should be 1, 10, 1, 1, 1.

General Information

Whenever a volume (disc) is formatted, all existing data is erased. To be sure the operator does indeed intend a CALL "FORMAT" command, the following message always appears after initiating CALL "FORMAT":

```
FORMAT REQUESTED, OK TO DESTROY DATA ON DEVICE 0?
```

Enter YES to complete the formatting procedure.

The amount of free (available) space remaining after a CALL "FORMAT" command may be determined by executing CALL "DSTAT".

This command automatically executes CALL "MOUNT" after its execution, but no device status message is generated.

The only way volume label information may be changed is by executing a CALL "DUP" command and copying all information to another disc.

Prerequisites

Device must be reserved.

Examples

Example 1:

```
CALL"FORMAT",2,"MEDCOSTS",1,1,"HOSP","PASS",1,10,1,1 1
```

This example accesses the disc in device 2 and specifies its name as "MEDCOSTS", its identification number as 1, the number of volumes involved as 1, the owner as "HOSP", and "PASS" as the master password. The number of 1st, 2nd, 3rd, 4th, and 5th level chains are 1,10,1,1,1, respectively.

Example 2:

```
CALL"FORMAT",A,A$,B,C,B$,C$,D,E,F,G,H
```

This example accesses the disc specified in A and specifies the identifying name located in A$. The identification volume number (which must be 1) is located in B, the number of volumes (which must be 1) is located in C, the owner's name is located in B$ and the master password located in C$. The number of 1st, 2nd, 3rd, 4th and 5th level chains are located in D, E, F, G and H, respectively.

CALL "HERRS" (Hard Error Status)

Purpose

The CALL "HERRS" command is a diagnostic tool only. This command allows a user to tell if the device (disc and drive) or the host is responsible for system malfunctions or slow downs. This command returns disc and drive hard and soft error information to the variables specified. Since CALL "HERRS" normally is used in diagnostic programs, it is not required in day-to-day system operation.

SYNTAX FORM

CAL "HERRS", numeric variable, numeric variable, numeric variable, numeric variable, numeric variable

DESCRIPTIVE FORM

CALL "HERRS", device address, no. of retries in last I/O, no. of accumulated retries, no. of successful I/O recoveries, no. of unsuccessful I/O operations

Field Definitions (Descriptive Form)

CALL "HERRS"

These are keywords for this command. Enter as shown in SYNTAX FORM.

device address

The device address is shown on or near the front of the device.

number of retries last I/O

This counter indicates the number of retries attempted during the last I/O operation, to a maximum of 255.

number of accumulated retries

This counter indicates the total number of retries since the last power up, to a maximum of 65,335.

number of successful I/O recoveries

This counter indicates the total number of I/O operations that have been successfully recovered after one or more retries, to a maximum count of 65,535. These are soft errors.

number of unsuccessful I/O operations

This counter indicates the total number of I/O operations that have not been successfully recovered, to a maximum count of 65,535. These are hard errors.

General Information

Although "HERRS" stands for "HARD ERROR STATUS", this command documents both hard and soft error frequency.

Whenever the system increments the hard error counter by one, an error message is displayed.

It is not necessary to execute the CALL "HERRS" command to initiate any of the counters, since they function independently. CALL "HERRS" is used only to access the information. The number of retries required before the hard error counter is updated may vary. See the 4907 Service Manual.

Prerequisites

None.

INIT (Initialize)

Purpose

This command resets all string variable, array variable, and numeric variable values to an undefined state.

This is a 4050 Series BASIC command. It is included here because it directly affects 4907 system operation.

SYNTAX FORM

```
INIT
```

DESCRIPTIVE FORM

```
INITIALIZE
```

General Information

Execution of this command closes all disc files, sets the current device to zero (0) and specifies "SCRATCHLIB" as the current library. INIT may be used to close sequential files which have been opened for full access (F) without losing any of the files.

Prerequisites

None.

INPUT

Purpose

The INPUT command reads ASCII data that has been stored in the designated file with a PRINT command. It operates like the 4050 Series BASIC INPUT command with the following exceptions:

- The lfn (logical file number) specified in the preceding OPEN command must be entered. This I/O address tells the system which disc file to access.
- Random, as well as sequential files, may be INPUT. This means that, unlike magnetic tape files, it is possible to directly access a particular location in the file.
- The primary address character is "#".

SYNTAX FORM

```
                                        (array variable  )  [ (array variable  ) ]
INP # constant [,numeric expression] :  {string variable  }  [,{string variable  } ]. . . .
                                        (numeric variable)  [ (numeric variable) ]
```

DESCRIPTIVE FORM

```
                                  target variables    [  target variables   ]
INPUT # lfn [,record number] :    for incoming data   [' for incoming data  ] . . . .
```

Field Definitions (Descriptive Form)

INPUT — This is the keyword for this command. It may be entered as shown in SYNTAX FORM.

lfn — Logical file number: This number must match the lfn specified in the OPEN command.

record number

An entry is necessary in this field if you are inputting from a random file. An entry of 1 or greater places the logical file pointer at the first string in the record of that number.

No entry is necessary in this field if you are inputting from a sequential file. However, if the statement is part of a program used to retrieve random file information one time and sequential file information the next, a numeric variable must be entered. See Example 4.

target numeric variables for incoming data

These entries specify where the string or parts of the string is to be sent.

General Information

- An INPUT command can only read ASCII information placed in the file with a PRINT command.
- For programming examples, see USING PRINT AND INPUT IN A PROGRAM (Appendix C).

NOTE

The End Of Record character for ASCII format files is a Carriage Return. Reaching the end of an ASCII record will not generate an interrupt (with respect to an ON EOF ... THEN ... statement).

Prerequisites

File must be open and in ASCII format.

Examples

Example 1: INPUTTING NUMERIC VALUES FROM A RANDOM FILE

```
330 INPUT #3,10:A,B,C
```

This example places the logical file pointer at the beginning of record 10 in the file associated with lfn 3. The system then finds the first numeric value in the first string. That value is stored in A. The system continues to search through record 10, string by string. If no further values are found, a search is begun through record 11. If none are found, the next record is searched and so on. (Remember, everything entered in the PRINT command up to the first Carriage Return is considered a single string. Everything between the first Carriage Return and the second Carriage Return is also considered a single string and so on.) Eventually, if a second and third value are located, they are stored in B and C.

If less than three values exist in the entire file, an error message appears.

Example 2: INPUTTING STRINGS FROM A RANDOM FILE

```
430 INPUT #3,12:A$
```

This example places the logical file pointer at the beginning of record 12 in the file with lfn 3. The system then reads everything in that record up to the first Carriage Return and stores it in A$.

Example 3: INPUTTING A STRING FROM A SEQUENTIAL FILE

```
410 INPUT #3:A$
```

INPUT will start at the current pointer position in lfn 3. The system then reads everything in the next string and stores it in A$.

Example 4: INPUTTING A STRING FROM A RANDOM OR SEQUENTIAL FILE

```
140 INPUT #4,A:A$
```

This retrieves a single string from the file and places it in A$. If A=0 and this is a sequential file, this command inputs the first string past the current pointer position.

If A=1 or greater and this is a random file, the string in the record (identified in A) is input.

KILL

Purpose

The KILL command deletes single files, groups of files or all files on a particular disc.

SYNTAX FORM

KIL "F.I." [,string]

DESCRIPTIVE FORM

KILL "F.I." [,master password]

Field Definitions (Descriptive Form)

KILL — This is the keyword for the command. It may be entered as shown in SYNTAX FORM.

F.I. — The file or files represented by the F.I. entry will all be deleted. The files must be closed and assigned passwords specified.

Any F.I. must be in quotes.

master password — If a master password was specified when the disc was originally formatted, it may be entered here. If the master password is entered, all closed files represented by the F.I. will be deleted. This will occur even if passwords normally required in the F.I. are not used.

A master password must be in quotes: "RED".

General Information

Libraries cannot be deleted unless the disc is reformatted. This is true even though all subsequent files may have been deleted.

This command will not be completed if more than eight files are open.

Prerequisites

Disc must be mounted but device must not be reserved. Files must be closed.

Examples

Example 1:

```
140 KILL "#","FRED"
```

This example deletes all closed files in the current library, whether or not the files or preceding libraries contain passwords. See SPECIAL CHARACTERS in Section 4 for a full description of the use of "#" and other special characters.

Example 2:

```
210 KILL "@MYLIBRY/A/B/C"
```

If file "C" is closed and contains no password, this example deletes it.

Example 3:

```
470 KILL "$A/B.*"
```

This example kills all closed "B" files contained in SYSLIB and library "A" when those files contain any kind of extension (including blank extensions).

CALL "MOUNT"

Purpose

The CALL "MOUNT" command tells the system that the disc is in place and may be used. This command also generates a device status message identical to that generated by the CALL "DSTAT" command. If CALL "MOUNT" is not executed, the system has no way of knowing if the disc is in place, that it is turned on, etc. The system must be informed with this command.

SYNTAX FORM

CAL "MOUNT", numeric expression, string variable

DESCRIPTIVE FORM

CALL "MOUNT", device address, target string variable

Field Definitions (Descriptive Form)

CALL "MOUNT"	These are the keywords for this command. Entry may be made as shown in SYNTAX FORM.
device address	The device address is shown on or near the front of the device.
target string variable	This is the location to which the device status message is sent. This string variable must first be dimensioned large enough to accommodate the incoming messages. Each device status message is 186 characters.

General Information

An error message appears if the device is not powered up, is not up to speed, or a CALL "SETTIM" command has not been executed.

Prerequisites

Set clock.

Examples

Example 1:

```
100 CALL "MOUNT",2,A$
```

This example tells the system the disc in device 2 is ready for use and sends the device status message to A$.

CALL "MRKBBG" (Mark Bad Block Group)

Purpose

The CALL "MRKBBG" command tells the system that there are defective areas on the disc that are not to be used. This command is necessary only after Error Message 15 appears specifying the address of defective tracks or sectors. CALL "MRKBBG" is executed with this address in the last field. The defective space will not be used for subsequent data storage. This command is not commonly used in a program.

SYNTAX FORM

CAL "MRKBBG", numeric expression, string, string, string

DESCRIPTIVE FORM

CALL "MRKBBG", device address, volume I.D., master password, address of defective space.

Field Definitions (Descriptive Form)

CALL "MRKBBG" — These are the keywords for this command. Enter as shown in SYNTAX FORM.

device address — The device address is shown on or near the front of the device.

volume I.D. — This entry must be the same as that specified during the CALL "FORMAT" or CALL "FFRMT" command. If the volume I.D. is unknown, a CALL "DSTAT" command displays this disc information. The volume I.D. must be entered in quotes: "TOM".

master password — The entry here must be the same as the entry in the CALL "FORMAT" or CALL "FFRMT" command. If there was no password but simply a null string ("") entered in that field, that string must also be entered here.

Any master password must be in quotes: "RED".

address of defective space

The entry in This field must match the first eight digits displayed in the last field of Error Message 15. This is the message that appears when device hard errors occur. These digits indicate the number and location of the blocks causing the errors.

The last two digits in the last field of Error Message 15 indicate whether the error is the result of mechanical or bad block problems. See General Information below.

The address must be in quotes.

General Information

To mark a bad block group follow this procedure:

1. Write down location of bad block as shown in error message.
2. Copy the disc using the DUP routine.
3. Close all files.
4. Reserve the device.
5. Delete bad file.
6. Execute CALL "MRKBBG" with location of bad block group.

If the CALL "MRKBBG" command is successfully executed but hard error messages continue, the problem is more likely one of hardware operation rather than defective areas on the disc. Even though Error Message 15 indicates the number and location of "bad blocks," the message may actually be reflecting a malfunctioning read/write head or similar problem. If execution of CALL "MRKBBG" does not end I/O problems, check the last two values in the error message (NN) against the list of specific problems in the 4907 File Manager Service Manual.

If volume I.D. or master password is entered incorrectly, the device must be dismounted and remounted before re-executing the command. This is necessary because the free space will show a decrease, but, in fact, the bad blocks will not have been marked.

If the volume I.D. or master password has been incorrectly entered in a CALL "MRKBBG" command, a subsequent CALL "CUSTAT" command will show that the free space has been decreased. The disc, however, will NOT be marked.

Prerequisites

Device must be reserved.

Example

The system has accessed device 1 with the volume I.D. of "SPECS." The master password is "LAB." The system is executing an I/O operation when this message appears:

```
ERROR 15 DEVICE I/O ERROR 010001FA-NN
```

After following the procedure in General Information, enter and execute the following:

```
CAL"MRKBBG",1,"SPECS","LAB","010001FA"
```

The bad blocks are marked, and no further attempts are made by the system to PRINT or WRITE data to them.

CALL "NEXT"

Purpose

The CALL "NEXT" command opens the next file in a series of files opened by an OPEN "G" (group) command. As each new file is opened, the CALL "NEXT" command generates a file status message which is sent to the target string variable.

SYNTAX FORM

CAL "NEXT", numeric expression, string variable

DESCRIPTIVE FORM

CALL "NEXT", lfn, target string variable

Field Definitions (Descriptive Form)

CALL "NEXT"

These are the keywords for the command. They may be entered as shown in SYNTAX FORM.

lfn

Logical file number. This number must be the same as specified in the previous OPEN command.

target string variable

As each new file is opened, its file status message is sent to this string variable. No status messages will be generated if a null string ("") is entered here instead of a string variable.

Any string variable used for file status messages must be dimensioned large enough to contain the information. Each full file status message is 189 characters plus the characters in the F.I.

General Information

- If a CALL "NEXT" command is executed after a normal OPEN command (not a group OPEN), an error message is displayed.
- If an OPEN "G" command locates a single file but the first CALL "NEXT" finds no files, a null string is generated and stored in the target string variable.
- If OPEN "G" locates no files, the first CALL "NEXT" generates an error message.
- If a CALL "NEXT" has located no files the following CALL "NEXT" generates an error message.
- When CALL "NEXT" is executed, the current file is closed.
- CALL "NEXT" always executes on a file in the current device. See UNIT command description.

Prerequisites

File must have been opened with an OPEN "G" command.

Examples

Example 1:

```
800 CALL"NEXT",2,A$
```

This example opens the next file in a group of files that were specified in an earlier OPEN "G" command as lfn 2. The file status message is sent to A$.

Example 2:

```
450 CALL"NEXT",A,B$
```

This example opens the next file in a group of files specified in an earlier OPEN "G" command. The lfn stored in A must be the same as that in the OPEN "G" command. The file status message is sent to B$.

OLD

Purpose

The OLD command loads a BASIC program in either binary or ASCII format from the disc file into the Graphic System memory.

SYNTAX FORM

OLD "F.I." [,string]

DESCRIPTIVE FORM

OLD "F.I." [,"A"]

Field Definitions (Descriptive Form)

OLD — This is the keyword for this command.

F.I. — This F.I. must match the F.I. used when the program was saved.

A — If the program was saved in ASCII form, this entry must appear. If the program was saved in binary form, no entry is necessary.

General Information

- An error message appears if the program is not found or is a data file.
- If the program was originally stored in binary, it must be retrieved in binary. The same applies to programs stored in ASCII.
- Because this command executes an automatic DELETE ALL command as part of the operation, the current library is set to "SCRATCHLIB." The current device is set to 0.
- If an ASCII file has been designated SECRET, it cannot be brought into memory with OLD. (A secret ASCII program may, however, be appended into memory.)

Prerequisites

Disc must be mounted but device must not be reserved. File does not have to be open.

Examples

Example 1:

```
140 OLD "@MYLIBRY/STOCKS"
```

This example transfers a binary program from the file "STOCKS" into Graphic System memory.

Example 2:

```
140 OLD "@MYLIBRY/STOCKS","A"
```

This example transfers an ASCII program from the file "STOCKS" into Graphic System memory.

ON EOF (On End-Of-File)

Purpose

This form of the 4050 Series BASIC ON . . . THEN statement allows continued program operation when the end-of-file is encountered. It is particularly useful when retrieving information from a sequential file of unknown length.

SYNTAX FORM

ON EOF (numeric expression) THE constant

DESCRIPTIVE FORM

ON EOF (lfn) THEN line number

Field Definitions (Descriptive Form)

ON EOF THEN — These are the keywords for the command.

lfn — Logical file number: This is the same number that is assigned to a particular file in the OPEN command.

line number — This line number is executed after encountering the end of the file.

General Information

- If no ON EOF command has been entered in the program, an error message is generated when the end-of-file has been reached.
- If the Graphic System is equipped with a C01 (Bisynchronous Interface), it will have the logical file number 9. This can cause a problem if a disc file has been assigned a logical file number 9. If both devices are in use in the same program, separate ON EOF commands must be provided for each.
- See the ON . . . THEN command description in the 4050 Series Graphic System Reference Manual for further details.

Prerequisites

None.

Example

```
120 ON EOF(2) THEN 140
```

This example executes line 140 when the end of file has been reached on the file represented by lfn 2.

OPEN

Purpose

The OPEN command must be executed before the majority of file management and I/O commands can be executed. See GENERAL SEQUENCE FLOW CHART in Section 2. This command allows the user to specify:

- A logical file number representing the file. This number eliminates the need for rewriting the F.I. in many subsequent commands.
- Whether access is to be read and write, read only, or update.
- Whether single or multiple files are to be opened.
- The string variable in which the file status message is to be stored.

SYNTAX FORM

OPE "F.I." [,"G"]; constant, "string", string variable

DESCRIPTIVE FORM

OPEN "F.I." ["GROUP"]; lfn, "type of access", target string variable

Field Definitions (Descriptive Form)

OPEN

This is the the keyword for the command. It may be entered as shown in SYNTAX FORM.

F.I.

This entry is the F.I. of the file to be opened. Special characters must be used when entering the F.I. in this field if multiple files are to be opened. See SPECIAL CHARACTERS in Section 4.

"GROUP"

This entry is required if multiple files are to be opened. Only the character "G" must be entered. File contents may be binary or ASCII in random or sequential files.

lfn

Logical file number: The number entered here must be an integer 1 to 9. This number is used to represent the F.I. in subsequent commands. This eliminates the need for writing out the F.I. each time.

type of access

This entry specifies the type of access intended for this file during this OPEN.

Enter an "R" if you intend to only read from a file.

Enter a "U" if updating (adding to) a SEQUENTIAL file. No data is destroyed if file is opened and rewound, partial data is entered, and the file is then closed.

Enter an "F" if writing to a RANDOM or SEQUENTIAL file. F indicates "FULL" access for read or write (all I/O operations).

This entry must be in quotes.

CAUTION

If a SEQUENTIAL file is opened for full access (F), the pointer is moved to the beginning of the file. If no reading or writing is carried out and the file is closed, the pointer remains at that point. This means that all data past that point is lost. This does not apply if files are closed with INIT or DELALL. If you have opened a sequential file for full access, you must READ or INPUT from the file (for example, store strings repeatedly into the same string variable) until the End Of File is reached. This moves the pointer to the end of the data, and the file can be closed.

To erase the contents of a sequential file, enter OPEN, then CLOSE, then OPEN.

target string variable

When the OPEN command is executed, a file status message is generated. This message is then stored in the string variable specified here. The parameters are:

- last date altered
- last date used
- date created
- number of current OPENS
- current bytes allocated
- number of bytes used
- record size
- file attributes

The last field in the message shows the entire F.I. for the file.

Remember to dimension the string variable large enough to contain the file message; that is, DIM A$(300). The file status message is 189 characters plus the characters in the F.I.

See DEVICE AND FILE STATUS MESSAGES (Appendix A) for further details.

If OPEN "G" has been executed, the file status message sent to the string shows the status of the first file to be opened. If there are no files, the string variable will contain a null string ("").

General Information

This command is executed on a file in the current device. See UNIT command description.

If no UNIT command is previously executed, the system defaults to Device 0.

If files are not created prior to an OPEN "G" command, an error message appears.

Occasionally it is necessary to review what has just been entered. To do this, execute the CALL REWIND command. This positions the file pointer back to the beginning of the file without CLOSE command and another OPEN command. See FILE POINTER OPERATION (Appendix C) for further details.

Hitting BREAK twice will also close files. However, this command also aborts program file control.

Prerequisites

Disc must be mounted but device must not be reserved.

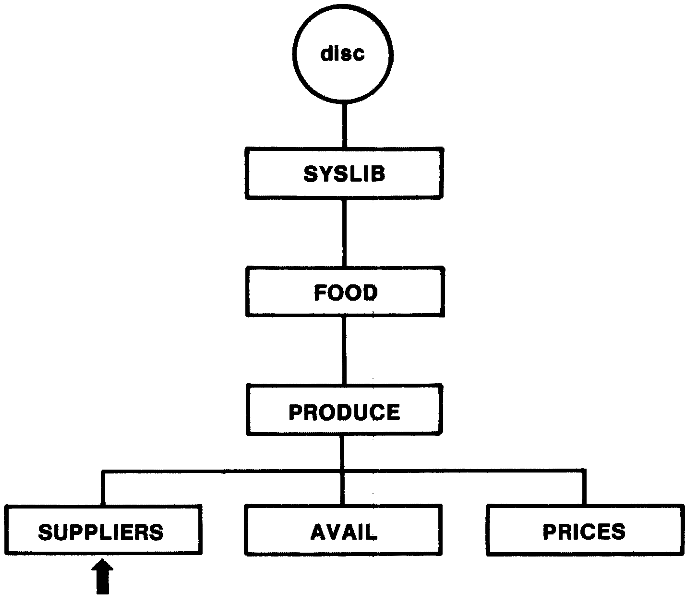
Examples

Example 1:

```
10 OPEN "$FOOD/PRODUCT/SUPPLIERS";3,"R",A$
```

This example accesses "SYSLIB" and libraries "FOOD" and "PRODUCE." It then opens the file "SUPPLIERS." The logical file number is specified as 3, the type of access is "R" (read only), and A$ is specified as the destination for the file status message.

The arrow indicates the file selected in the SYSLIB storage structure shown in Figure 5-1.



2380-34

Figure 5-1. Sample Storage Structure (4 Levels, 3 Files).

Example 2:

```
115 OPEN A$;5,B$,C$
```

This example OPEN the file specified by the F.I. in A$. The lfn is specified as 5, the character in B$ specifies the type of file access, and C$ is specified as the destination for the file status message.

Example 3:

```
350 OPEN "@ MYLIBRY/HOSPITAL/STJOHNS
        #.RES","G";3,"F",B$
```

This example accesses "MYLIBRY," "HOSPITAL" and "STJOHNS," and all subsequent files (#) with the extension "RES". Since a "G" is specified in the next field, this is a "GROUP" OPEN. If anything other than "G" is entered, an error message appears.

The lfn is specified as 3, the type of file access is "F" (for FULL), and B$ is the destination for the file status message.

The arrows indicate the files selected in the "MYLIBRY" storage structure shown in Figure 5-2.

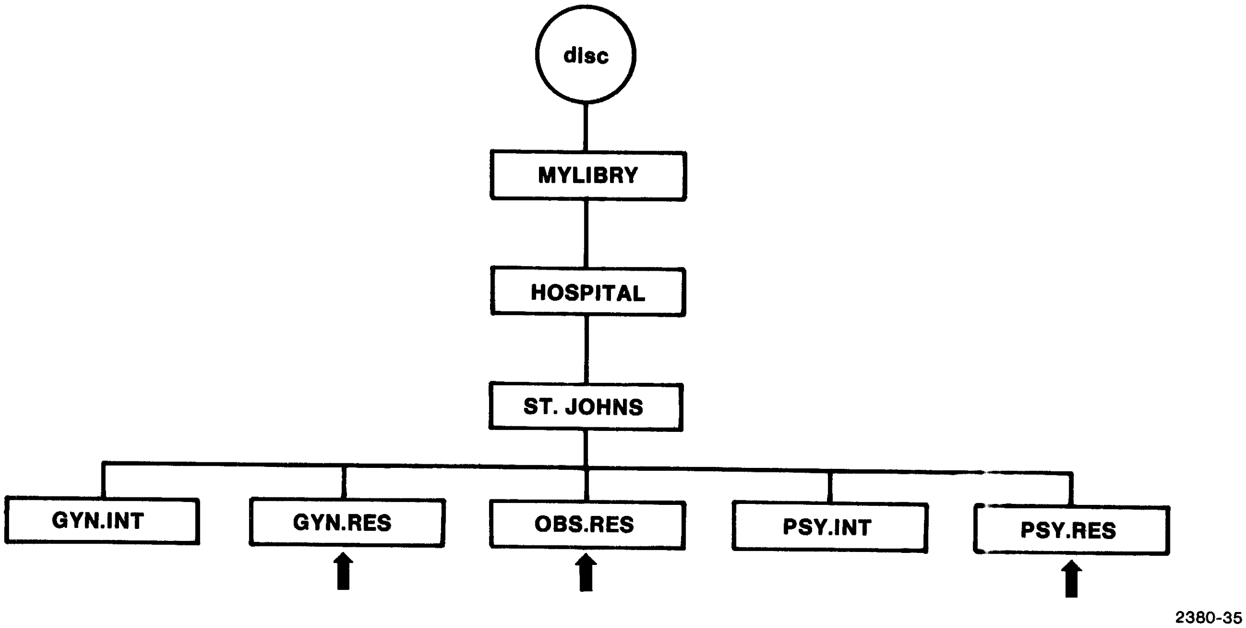


Figure 5-2. Sample Storage Structure (4 Levels, 5 Files).

Opening Multiple Files. The following sample shows how multiple files on different devices may be opened, assuming there are two devices, with addresses 0 and 1.

```
100 UNIT 0
110 OPEN A$;2,B$,C$
120 UNIT 1
130 OPEN D$;3,E$,F$
140 (various I/O operations to lfn 2 and 3)
```

PRINT

Purpose

The PRINT command stores ASCII data in the designated file. It operates like the 4050 Series BASIC PRINT command with the following exceptions:

- The lfn (logical file number) specified in the preceding OPEN command must be entered. This I/O address tells the system which disc file to access.

- Random as well as sequential files may be printed. This means that, unlike magnetic tape files, it is possible to directly access a particular location in the file.

- The primary address character is "#".

SYNTAX FORM

```
                                       ┌    ⎧"string" ⎫ ┐ ┌⎧numeric expression⎫
PRI # constant [,numeric expression] : │USI⎨   or    ⎬:│ │⎨                  ⎬
                                       └    ⎩constant ⎭ ┘ └⎩string            ⎭
┌⎧ , ⎫ ┌ ⎧numeric expression⎫┐┐
│⎨   ⎬ │,⎨                  ⎬││ ....  [;]
└⎩ ; ⎭ └ ⎩string            ⎭┘┘
```

DESCRIPTIVE FORM

```
                                    ⎧"format string"   ⎫
PRINT # lfn [,record number] : [USING ⎨      or          ⎬ :][item to be printed
                                    ⎩IMAGE line number ⎭
[{ , } item to be printed ]] ... [;]
   ;
```

Field Definitions (Descriptive Form)

PRINT — This is the keyword for this command. It may be entered as shown in SYNTAX FORM.

lfn — Logical file number: The number entered here must match the lfn specified in the OPEN command.

record number An entry is necessary in this field if you are printing to a random file. The value of this entry must be 1 or greater. The logical file pointer is placed at the beginning of the record specified.

No entry is necessary in this field if you are printing to a sequential file. However, if the statement is part of a program and is used to enter information in a sequential file one time and a random file another time, a numeric variable must be entered.

USING See the 4050 Series Graphic System Reference Manual.

data item This is the item to be stored. It may be the actual constant or string constant, a numeric expression, the location of the item in memory (A, A$, etc.), or a combination.

data item Same as previous field.

General Information

- If you are updating a sequential file (that is, adding data to the end of existing information), "U" must be entered in the OPEN command. This places the logical file pointer at the end of the last item. PRINT then adds new data beginning at that point.
- Printing data to a record requires that all previous records be filled. See the file initializing program in SAMPLE PROGRAMS (Appendix F).
- If too much data is printed to a record in a random file the excess is placed in one or more following records. This destroys any existing data in those records.
- For examples of how to retrieve PRINTed information, see INPUT command description.
- For programming examples, see USING PRINT AND INPUT IN A PROGRAM (Appendix C).

Prerequisites

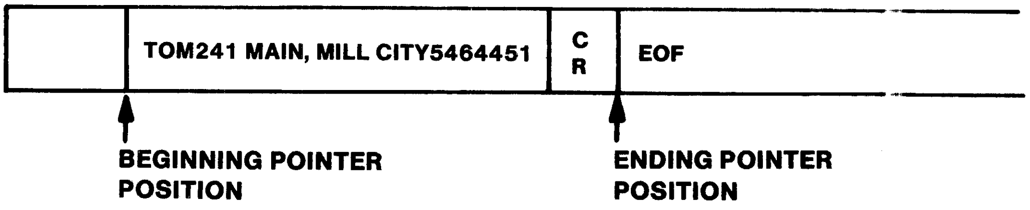
Disc must be mounted but device must not be reserved. File must be open and in ASCII format.

Examples

Example 1: PRINTING TO A SEQUENTIAL FILE

```
140 PRINT #3:"TOM"; "241 MAIN, MILL CITY";5464451
```

The system considers TOM, 241 MAIN, MILL CITY 5464451 as a single string and stores it, starting at the current pointer position. After PRINT is executed, the pointer is located just past the CR (carriage return) at the end of the string.



Another string may be added by executing another PRINT command. The new information is recorded where the logical file pointer is located. The EOF mark is always moved along and placed just after the last CR.

Example 2: PRINTING TO A SEQUENTIAL FILE

```
135 PRINT #3: A$
```

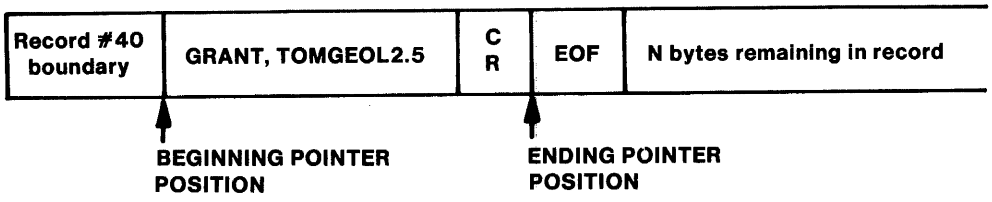
This example functions exactly as Example 1 if:

A$ = "TOM 241 MAIN, MILL CITY 5464451"

Example 3: PRINTING TO A RANDOM FILE

```
340 PRINT #3,40:"GRANT, TOM";"GEOL";2.5
```

The system considers GRANT, TOM, GEOL and 2.5 as a single string and stores it that way at the very beginning of record 40 in lfn 3. After PRINT is executed, the logical file pointer is located just past the CR at the end of the string.



Even if there is enough space left in the record, no additional strings may be added by executing another PRINT command; each additional PRINT starts at the beginning of a record.

CAUTION

Printing to a random ASCII file must be done with great accuracy. If more data is transmitted than can be accommodated in the specified record, the data in the following record is overwritten.

READ

Purpose

The READ command reads binary data stored in the designated file with a WRITE command. It operates like the 4050 Series BASIC READ command with the following exceptions:

- The lfn (logical file number) specified in the preceding OPEN command must be entered. This I/O address tells the system which disc file to access.
- Random as well as sequential files may be read. This means that, unlike magnetic tape files, it is possible to directly access a particular location in the file.
- The primary address character is "#".

SYNTAX FORM

```
                                       (array variable  )  [  (array variable  ) ]
REA # constant [,numeric expression] : {string variable  }  [ ,{string variable  } ] ...
                                       (numeric variable)  [  (numeric variable) ]
```

DESCRIPTIVE FORM

```
                                target variables    [   target variables  ]
READ # lfn  [,record number] :  for incoming data   [ , for incoming data ] ...
```

Field Definitions (Descriptive Form)

READ

This is the keyword for this command. The entry may be made as shown in SYNTAX FORM.

lfn

Logical file number: The number must match the lfn specified in the OPEN command.

record number

An entry is necessary in this field only if you are reading from a random file. An entry of 1 or greater places the logical file pointer at the first item in the record of that number. The READ starts from there.

No entry is necessary in this field if you are reading from a sequential file. However, if this is part of a program used to read random files one time, and sequential files the next, a numeric variable must be entered. See Example 4.

target variables for incoming data — These entries specify where the item (or items) being read is to be sent. See the 4050 Series Graphic System Reference Manual for further details on array variables.

General Information

A READ command can only retrieve binary information placed in the file with a WRITE.

For programming examples, see USING WRITE AND READ IN A PROGRAM (Appendix C).

Prerequisites

Disc must be mounted but device must not be reserved. File must be open and in binary format.

Examples

Example 1: READING NUMERIC ITEMS FROM A RANDOM FILE

```
450 READ #3, 10:A,B,C
```

This example places the logical file pointer at the first item in record 10 in lfn 3, reads the first three items, which must all be numeric, and stores them in the variables shown. The record number in this statement shows that this is a random file. Unlike an INPUT command, a READ command stops looking when it encounters an EOR (End-of-Record). If the three items have not yet been found, an error message is generated.

Example 2: READING NUMERIC ITEMS FROM A SEQUENTIAL FILE

```
610 READ #3:A,B,C
```

Because this is a sequential file (no record number entered), READ starts at the current pointer position. The first three items past this position must be numeric values. They are read and then stored in the variables shown.

Example 3: READING STRINGS AND NUMERIC ITEMS FROM A RANDOM FILE

```
740 READ#1,2:A$,B$,C
```

This example reads the first three items in record 2 in lfn 1. These items, which must be a string, a string, and a variable, in that order, are then stored in the target variables shown.

Example 4: READING STRINGS AND NUMERIC ITEMS FROM A RANDOM OR SEQUENTIAL FILE

```
190 READ #5,B:A$,B$,C
```

This reads the first three items.

If B = 0 and the file is sequential, the system reads the first three items past the logical pointer position and places them in the variables shown.

If B = 1 or greater and the file is random, the system reads the first three items in the record of that number and places them in the variables shown.

The first three items must be a string, a string, and a numeric value.

CALL "RENAME"

Purpose

The CALL "RENAME" command is used to change the names of files, libraries, passwords, and extensions in F.I.'s. In some instances, changing names results in moving files to another library. This command is not generally used in a program.

SYNTAX FORM

CAL "RENAME", numeric expression, F.I., F.I.

DESCRIPTIVE FORM

CALL "RENAME", device address, old F.I., new F.I.

Field Definitions (Descriptive Form)

CALL "RENAME" These are the keywords for this command. Entry may be made as shown in SYNTAX FORM.

device address The device address may be seen on or near the front of the device.

old F.I. The F.I. of the file to be renamed is entered here. Special characters may be used.

The F.I. must be in quotes.

new F.I. The new F.I. is entered here.

The F.I. must be in quotes.

General Information

The CALL "RENAME" command works only on those files with a creation date earlier than the present date of the system clock. For example, if you wish to rename a file that has an inaccurate creation date of February 10, 1987, and the correct time of the system clock is July 14, 1978, CALL "RENAME" will fail. In fact, if the file creation date is only one second ahead of the system clock, the command will fail. Therefore, the correct date and time must always be used in the "SETTIM" routine.

This command operates much like COPY . . . TO when the command is used to change the names of libraries, files, passwords or instructions. There are two exceptions:

- The CALL "RENAME" command will not copy files to another device.
- When a file is moved using CALL "RENAME", the file in the original location is erased and not retained as it is in COPY . . . TO.

This command will not be completed if more than eight files are open.

Prerequisites

Disc must be mounted but device must not be reserved.

Examples

Example 1:

```
CALL"RENAME",2,"@YOURLIB/TOM/JANE","@YOURLIB/TOM/BILL"
```

This example changes the name of the file in the library "TOM" in YOURLIB from "JANE" to "BILL."

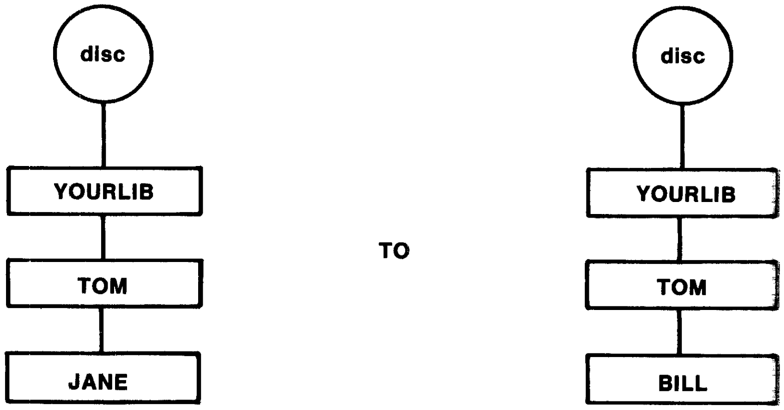


Figure 5-3. Sample Storage Structure Illustrating How to Change File Names.

Example 2:

```
CALL"RENAME",2,"@YOURLIB/TOM/JANE","$TOM/JANE"
```

This example transfers the library "TOM" and the file "JANE" from YOURLIB to SYSLIB, as well as changing the F.I.

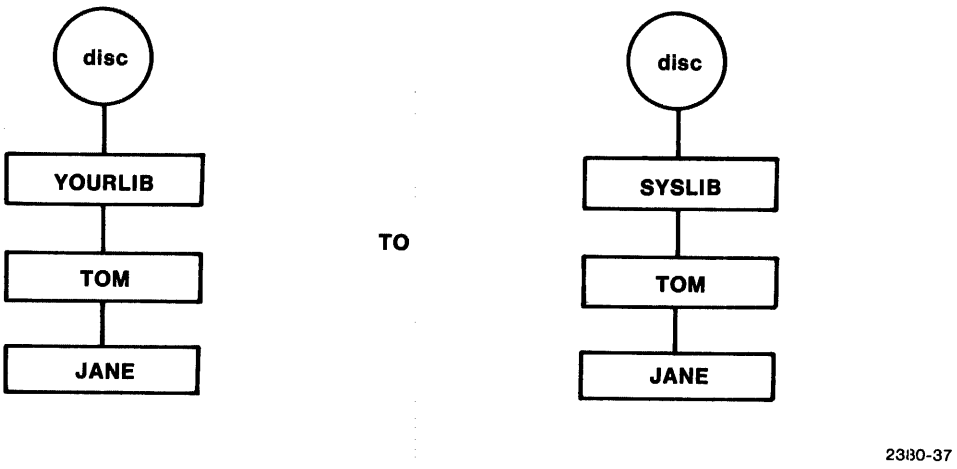


Figure 5-4. Sample Storage Structure Illustrating How to Transfer Files From One Library to Another.

Example 3:

```
CALL"RENAME",A,A$,B$
```

This example accesses the device with its address in A and changes the F.I. in A$ to the F.I. in B$.

Multiple File Name Changing. Special characters (#, *, ?) are required to change more than one F.I. with a single CALL "RENAME" command. See CHANGING NAMES in the COPY . . . TO command description.

CALL "REWIND"

Purpose

The CALL "REWIND" command allows the pointer to be positioned back to the beginning of a sequential file (after the header). This allows data just entered to be read or new data to be reentered without the usual CLOSE and subsequent OPEN command. This command is unnecessary for random access files since commands with the record number one (1) will accomplish the same thing.

SYNTAX FORM

CAL "REWIND", numeric expression

DESCRIPTIVE FORM

CALL "REWIND", lfn

Field Definitions (Descriptive Form)

CALL "REWIND"	These are the keywords for the command. Entry may be made as shown in SYNTAX FORM.
lfn	This entry must be the same as specified in the original OPEN command for this file.

General Information

See FILE POINTER OPERATION (Appendix C) for further details.

This command is always executed on a file in the current device. See UNIT command description.

Prerequisites

File must be open.

Example

```
400 CALL "REWIND",2
```

This example rewinds the pointer to the beginning of the file identified by lfn 2.

SAVE

Purpose

The SAVE command transfers a copy of the BASIC program currently in memory to a file on the disc. The file can be in either binary or ASCII form. A SAVE command will create its own file if necessary.

SYNTAX FORM

SAV “F.I.” [,string]; [, constant [, constant]]

DESCRIPTIVE FORM

SAVE “F.I.” [,“A”]; [Line number [,beginning, ending line number]]

Field Definitions (Descriptive Form)

SAVE

This is the keyword for the command. It may be entered as shown in SYNTAX FORM.

F.I.

This F.I. must be a valid F.I. (see Section 4). If no file has been created, this command will create its own.

Any F.I. must be in quotes.

A

If “A” is entered in this field, the program is sent to storage in ASCII form. If no entry is made, the program is stored in host binary form.

“A” must be in quotes.

beginning line number

If no ending line number is entered, a single line, specified by this entry, is saved.

ending line number

No entry can be made here unless a beginning line number is also entered. The program transfer ends after the statement with this line number.

General Information

- The stored program may be retrieved with an APPEND or OLD command.
- If speed is important, programs should be saved in binary (default) form.
- A SAVE command may not be executed to a random file.
- A file does not need to be opened to execute a SAVE command.
- If the file exists prior to this command, the file type (ASCII or binary) must match the type specified in the command. If the file does not exist, a new file of the correct type, with the Private attribute and minimum length to store the program, is automatically created.

Prerequisites

Disc must be mounted but device must not be reserved. File does not have to be open.

Examples

Example 1:

```
400 SAVE "@MYLIBRY/STOCKS";100,500
```

The statements from 100 to 500 of the current program in memory are stored in a file with the F.I. shown. The program is stored in binary format. The resulting file has the host binary attribute (H).

Example 2:

```
350 SAVE A$,"A"
```

This example places the entire program in memory into the file that has its F.I. stored in A$. The program is saved in ASCII format.

SECRET

Purpose

The SECRET command prevents future listing of a binary program. It is executed just prior to saving the program to a file.

SYNTAX FORM

```
SEC
```

DESCRIPTIVE FORM

```
SECRET
```

Field Definitions (Descriptive Form)

SECRET — This is the keyword for this command. Entry as shown in SYNTAX FORM.

General Information

If this command is written to an ASCII program, the program cannot be brought into Graphic System memory with an OLD command. ASCII programs with a SECRET designation, however, may be appended into memory.

This command may be executed on a magnetic tape or disc file.

Prerequisites

None.

CALL "SETTIM" (Set Time)

Purpose

The CALL "SETTIM" command sets the system clock so the exact time and date may be recorded when files are created, used or altered. The system clock is also necessary to mark the exact time and date on the volume label when discs are formatted.

SYNTAX FORM

```
CAL "SETTIM", string
```

DESCRIPTIVE FORM

```
CALL "SETTIM", date and time
```

Field Definitions (Descriptive Form)

CALL "SETTIM"	These are the keywords for this command. The entry may be made as shown in SYNTAX FORM.
date and time	The format for this field is: "DAY-MONTH-YEAR HOUR:MINUTES:SECONDS" Except for "month," which requires three characters, all subfields in this field may be one or two characters long. "DD-MON-YY HH:MM:SS" For example, if the system clock is to be set at one minute and 20 seconds past two PM on the fifth of January 1978, the entry is written this way: "5-JAN-78 14:1:20" Note that the "time" subfields are based on the military time scale of 000 to 2400 hours and that the entire entry is in quotes. The syntax for this range is 0:0:0 to 23:59:59.

General Information

NOTE

> *The system clock must be operating whenever the system is being used. When the clock has not been set, the clock indicator on the controller front panel is lit. The system clock must be set whenever the system is powered up. The time entered in the CALL "SETTIM" command must be accurate or certain commands such as CALL "RENAME" will not work correctly.*

The "seconds" subfield is optional. All characters entered past the seconds subfield are ignored.

The first three characters of the word for that month must be entered in the month field.

Entries must be made in both date **and** time fields. The command cannot enter dates later than 2040 A.D.

The system generates a February 29 on leap years. On non-leap years, a December 32 is generated.

Prerequisites

None.

Examples

```
CALL"SETTIM","14-DEC-77 20:30"
```

This example sets the system clock at thirty minutes past eight in the evening, December 14, 1977.

```
CALL"SETTIM",A$
```

This example sets the system clock to whatever date/time specifications are stored in A$.

CALL "SPACE"

Purpose

The CALL "SPACE" command is used to reduce or increase allocated file space. Normally, however, it is used to reduce the allocated space for a file only partially filled so that more space is available for other files. Although it may be used to increase a file size the system's dynamic space allocation feature makes this unnecessary.

SYNTAX FORM

CAL "SPACE", numeric expression, numeric expression, numeric variable,
numeric variable

DESCRIPTIVE FORM

CALL "SPACE", lfn, requested file size, target numeric variable,
target numeric variable

Field Definitions (Descriptive Form)

CALL "SPACE"

This is the keyword for this command. Entry may be made as shown in SYNTAX FORM.

lfn

Logical file number: This entry must match the lfn specified in the OPEN command.

desired file size

This entry, in bytes specifies how large the file is to be. The actual number of bytes will be in increments of 256.

target numeric variable

The amount of data, in bytes, already in the file is stored in the variable entered here.

target numeric variable

The actual file size, in bytes, after command execution is stored in the variable entered here.

General Information

The space available for increasing a file may be seen by executing a CALL "DSTAT" command; this command returns the number of free bytes remaining on a disc.

If the number of bytes specified in "desired file size" is less than the data required, the file will be reduced to just enough to hold the data.

An easy way to release unused file space on the disc is to execute CALL "SPACE" with zero (0) bytes requested.

Prerequisites

File must be open.

Example

```
150 CALL "SPACE",3,1000,A,B
```

This example allocates 1024 bytes to the file associated with lfn 3. The number of bytes required by the data already in the file is sent to A. The actual number of bytes in the file after command execution is sent to B.

CALL "TIME"

Purpose

The CALL "TIME" command returns the current date and time, according to the system clock, to the string variable specified in the last field.

SYNTAX FORM

CAL "TIME", string variable

DESCRIPTIVE FORM

CALL "TIME", target string variable

Field Definitions (Descriptive Form)

CALL "TIME" — These are the keywords for this command. They may be entered as shown in SYNTAX FORM.

target string variable — This is the location to which the 18 character time "message" is sent.

General Information

If the system clock is not operating, a null string is sent to the return string variable.

Prerequisites

A previous execution of CALL "SETTIM" is necessary for an accurate message.

Example

```
CAL "TIME", A$
```

This example sends the current date and time, according to the system clock, to A$.

TYP (Type)

Purpose

TYP is a function rather than a command and performs in the same way for disc files as it does for the magnetic tape files. TYP determines whether the data is in ASCII or binary form, and if binary, whether data is numeric or alphanumeric. It also establishes if the file is empty, open, and whether the end of the data has been reached (EOF character) or is illegal.

This function is necessary whenever the type of data is unknown. It returns an integer from 0 to 5, indicating one of the above conditions. See Table 5-1. See TYP in the 4050 Series Graphic System Reference Manual for details.

SYNTAX FORM

TYP (numeric expression)

DESCRIPTIVE FORM

TYPE (lfn)

Field Definitions (Descriptive Form)

TYPE This is the keyword for the function. Enter as shown in the SYNTAX FORM.

lfn Logical file number: This is the same number specified in the OPEN command.

See TYP in the 4050 Series Graphic System Reference Manual for further details.

Table 5-1

DATA TYPE TABLE

0	Empty File or File Not Open
1	Pointer is at End of File
2	Numeric Data or Character String Data/ASCII Format
3	Numeric Data/Binary Format
4	Character String Data/Binary Format
5	Illegal Data

If the integer 5 is returned, data is illegal and cannot be read.

Prerequisites

File must be open.

UNIT

Purpose

The UNIT command specifies the current device. Commands containing FLs or lfns without device addresses are directed to the current device. If no UNIT command is executed, the system designates device zero (0) as the current device.

SYNTAX FORM

UNI constant

DESCRIPTIVE FORM

UNIT device address

Field Definitions (Descriptive Form)

UNIT — This is the keyword. It may be entered as shown in SYNTAX FORM.

device address — The device address may be seen on or near the front of the device.

General Information

The UNIT command must be executed each time the system is powered up if any device other than 0 is to be the current device.

The following commands always attempt to execute on the current device specified by the UNIT command:

APPEND
ASSIGN
CALL "NEXT"
CALL "REWIND"
CLOSE
CREATE
DIRECTORY
INPUT
KILL
OLD
OPEN
PRINT
READ
SAVE
WRITE

If no UNIT command is executed, device zero (0) is considered the current device.

NOTE

The INIT, OLD and DELALL commands also reset the current device address to zero (0).

Device zero (0) may or may not exist, depending on your system. This depends on the strapping procedures carried out when your system was installed. Your Tektronix service person will identify the device addresses after installation.

Prerequisites

None.

Example

```
110 UNIT 2
```

This example specifies device 2 as the current device.

CALL "UNIT"

Purpose

The CALL "UNIT" command specifies the current device. Commands containing F.I.s or lfns but no device addresses are directed to the current device. If no CALL "UNIT" command is executed, the system designates device zero (0) as the current device.

SYNTAX FORM

CAL "UNIT", numeric expression

DESCRIPTIVE FORM

CALL "UNIT", device address

NOTE

This command functions exactly as the UNIT command. The only difference is that UNIT requires a constant in the device address field where CALL "UNIT" allows a numeric expression.

Field Definitions (Descriptive Form)

CALL "UNIT" — These are the keywords. They may be entered as shown in SYNTAX FORM.

device address — The device address may be seen on or near the front of the device.

General Information

The CALL "UNIT" command must be executed each time the system is powered up, if any device other than 0 is to be the current device.

The following commands always attempt to execute on the current device specified by the CALL "UNIT" command:

APPEND
ASSIGN
CALL "NEXT"
CALL "REWIND"
CLOSE
CREATE
DIRECTORY
INPUT
KILL
OLD
OPEN
PRINT
READ
SAVE
WRITE

If no CALL "UNIT" command is executed, device zero (0) is considered the current device.

The INIT, OLD and DELALL commands also reset the current device to zero.

Device zero (0) may or may not exist, depending on your system. This depends on the strapping procedures carried out when your system was installed.

Prerequisites

None.

Example

```
120 CALL"UNIT",B
```

This example specifies the device whose address is in B as the current device.

CALL "USERLIB"

Purpose

The CALL "USERLIB" command specifies the "current library." This command makes repeated file access more convenient.

The CALL "USERLIB" command places a portion of an F.I. in memory where it may automatically be recalled to precede the balance of the F.I. in subsequent commands. This portion is called the current library.

SYNTAX FORM

CAL "USERLIB", "string"

DESCRIPTIVE FORM

CALL "USERLIB", partial "F.I."

Field Definitions (Descriptive Form)

CALL "USERLIB" These are the keywords for this command. They may be entered as shown in SYNTAX FORM.

partial F.I. This entry may match up to the first 21 characters of the F.I. of the file to be accessed. All delimiters, (/), (:), (.), as well as passwords, must be included in the 21 character count.

Names on only the first four of the five available storage levels may be entered in this field.

If a string variable is entered in this field, it must also represent 21 characters or less.

General Information

A slash is automatically assigned to the end of the partial F.I. entered in this command. Even though it may be considered as a 22nd character, it is accepted by the system.

The current library specified in a CALL "USERLIB" command may be circumvented by preceding an F.I. with a commercial "at" sign (@) or a dollar sign ($). See SPECIAL CHARACTERS in Section 4.

Special characters (#, *, ?) should not be used in a CALL "USERLIB" command.

END, INIT, OLD or DELALL commands disable a previous CALL "USERLIB" command. These commands automatically specify SCRATCHLIB as the current library.

If a null string is entered with a CALL "USERLIB" (CALL "USERLIB",""), then there is no current library.

Prerequisites

None.

Examples

Example 1:

If your disc just contains files, then the F.I. used to access any file will consist of the name of the file only; i.e.: "TAX". However, you must circumvent or disable the current library provided by the system. To circumvent this library enter (@) with the file name, i.e. "TAX". To disable the current library enter:

```
CALL "USERLIB",""
```

Example 2:

If several 4th level files below the same 3rd level library (PROJ) are to be repeatedly accessed, the names of the first three levels may be entered in a CALL "USERLIB" command. For example, if the F.I. of one of the files is

"MYLIBRY/PROG/PROJ/DATA,"

then everything up to the name of the file, except the last slash (/), may be placed in a CALL "USERLIB" command in this way:

```
110 CALL "USERLIB", "MYLIBRY/PROG/PROJ"
```

The only entry repeatedly required in the command accessing the file is "DATA".

Example 3:

If many libraries and files under one particular 1st level library are to be accessed, only the 1st level library name can be part of a CALL "USERLIB" command. For example, if two F.I.'s used in a program look like this:

"PRODUCER/PRICES/LIST"

"PRODUCER/SALESMEN/AREA1"

Then the CALL "USERLIB" command can only contain "PRODUCER" as shown below.

```
150 CALL "USERLIB","PRODUCER"
```

The F.I.'s used throughout the balance of the program must be written like this:

"PRICES/LIST"

or this:

"SALESMEN/AREA1"

WRITE

Purpose

The WRITE command enters binary data into the designated file. It operates like the 4051 BASIC WRITE command with the following exceptions:

- The lfn (logical file number) specified in the preceding OPEN command must be entered. This I/O address tells the system which disc file to access.
- Random as well as sequential files may be written. This means that, unlike magnetic tape files, it is possible to directly access a particular location in the file when writing or reading from a file.
- The primary address character is "#".

Unlike entries in the PRINT command, each string or numeric value in a WRITE command is considered a separate entry.

SYNTAX FORM

$$\text{WRI \# constant [,numeric expression] : }\left\{\begin{matrix}\text{numeric expression}\\ \text{string variable}\end{matrix}\right\}\left[,\left\{\begin{matrix}\text{numeric expression}\\ \text{string variable}\end{matrix}\right\}\right]\ldots$$

DESCRIPTIVE FORM

WRITE # lfn [,record number] : data item [,data item] . . .

Field Definitions (Descriptive Form)

WRITE — This is the keyword for this command. The entry may be made as shown in SYNTAX FORM.

lfn — Logical file number: The number here must match the lfn specified in the OPEN command.

record number

An entry is necessary here if you are writing to a random file.

The logical file pointer is placed at the beginning of the record specified. That is where WRITE will begin. When WRITE is executed, all subsequent items in THAT RECORD ONLY are erased.

No entry is necessary if you are writing to a sequential file. If the WRITE is part of a program and is used to enter information in a sequential file one time and a random file another time, a variable must be entered. See Example 4.

data item

This is the item to be stored. It may be the actual item (constant), a numeric expression, a variable (A, A$, etc.), or a combination.

A constant must be in quotes.

data item

Same as previous field.

General Information

- If you are "updating" a sequential file, that is, adding data to the end of existing information, "U" must be entered in the previous OPEN command. This places the logical file pointer at the end of the last item. WRITE then adds data beginning at the pointer location.
- For programming examples see USING WRITE AND READ IN A PROGRAM (Appendix C).

Writing Data "In Between" In Sequential Files. As we have described, the logical file pointer in sequential files is always placed at the beginning of the file or at the end of the last item (with "U" in the OPEN command). There are times, however, when it may be necessary to WRITE data at a point between the first and last items. This means alternating TYP functions[1] and READ commands must be executed. This will move the pointer item by item until it is at the desired location. WRITE may then be executed. Remember that all existing items past that point to the end of the file will be erased.

This procedure is not possible with random files because each WRITE command returns the logical file pointer to the beginning of the record. See READ command description for examples of how to retrieve information written to a file.

[1] **TYP functions are necessary only if the type of data item (a string or number) is unknown.**

Prerequisites

Disc must be mounted but device must not be reserved. File must be open and in binary format.

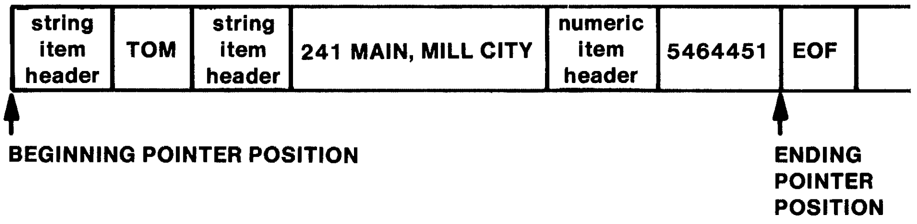
Examples

Example 1: WRITING STRINGS AND A NUMERIC VALUE TO A SEQUENTIAL FILE

```
150 WRITE #3:"TOM","241 MAIN, MILL CITY",5464451
```

This example stores "TOM", "241 MAIN, MILL CITY" and 5464451 as three separate items: a string, a string, and a numeric value.

Remember, any number within quotes is not considered a numeric value but part of a string. The WRITE command starts the at current pointer position. After the WRITE command is executed, the pointer is located at the end of the third item.



Another WRITE command may be executed to add more strings. The new information is recorded starting where the logical pointer is located. If the file is closed first and then reopened, the next WRITE command places the new data at the beginning of the file. The OPEN command must contain "U", which will start WRITE at the end of the existing data.

Example 2: WRITING STRINGS AND A NUMERIC VALUE TO A SEQUENTIAL FILE

```
160 WRITE #3: A$, B$, C
```

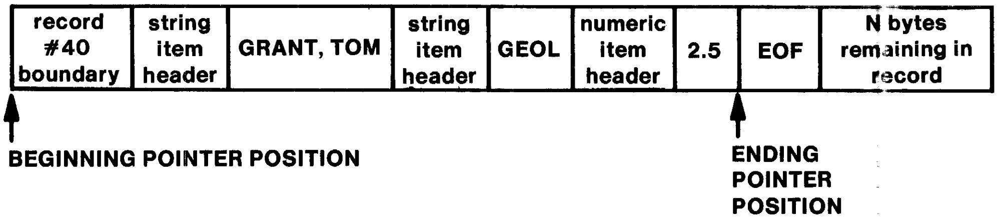
This example functions exactly as Example 2 if:

A$ = "TOM"
B$ = "241 MAIN, MILL CITY"
C = 5464451

Example 3: WRITING STRINGS AND A NUMERIC VALUE TO A RANDOM FILE

```
340 WRITE #3, 40: "GRANT, TOM", "GEOL", 2.5
```

This example stores "GRANT", "TOM", "GEOL", and 2.5 as three separate items: a string, a string, and a number. The items are placed at the very beginning of record 40 in lfn 3.



The logical pointer is now located at the end of the third item.

NOTE

If the record is too short for the binary information to be entered, an error message appears.

If another WRITE command is executed to this record, the logical file pointer will move back, and the new information will be placed at the beginning of record 40.

If information exceeds the remaining record space, an error message will appear. If it is less, the gap between the last item and the next record boundary will be ignored.

Example 4: WRITING TO A RANDOM OR SEQUENTIAL FILE

```
260 WRITE #3,A:"TOM",61,2.45
```

This example writes the items "TOM", 61, 2.45 to lfn 3. If the value of A is zero (0), the system assumes the file is sequential. If the value of A is 1 or greater, it assumes the file is random and starts writing at the record with that number.

Example 5: WRITING A SINGLE STRING TO A RANDOM FILE

```
110 WRITE #3,20:"SMITH, 2.51"
```

This example writes a single string item to the beginning of record 20 in lfn 3.

SECTION 6

CONTENTS

Section 6

ROUTINE MAINTENANCE

The table below shows routine maintenance for the flexible disc drives. Procedures 1 and 2 may be performed by the operator. Procedures 3 through 6 should be carried out by a Tektronix serviceman and are fully described in the Flexible Disc Drive Service Manual.

Procedure	Item	Inspect For	Interval	Action Required
1	Read/write head	Oxide build-up resulting in repeated hard or soft errors	12 months	Clean read/write head.
2	Read/write head	Worn felt	12 months	Replace button.
3	Stepper motor and lead screw	Nicks, burrs, and dirt	12 months	Clean off oil, dust, and dirt. Dress down nicks or burrs, or replace part.
4	Belt	Frayed or weak areas	12 months	Replace.
5	Base	Loose screws, switches, and connectors	12 months	Tighten screws, connectors, and switches.
		Check for dust and dirt		Clean off dust and dirt.
6	Read/write head	Aborted I/O commands or distorted results	12 months	Align head.

Procedure No. 1

a. Remove cover from drive (Figure 6-1).

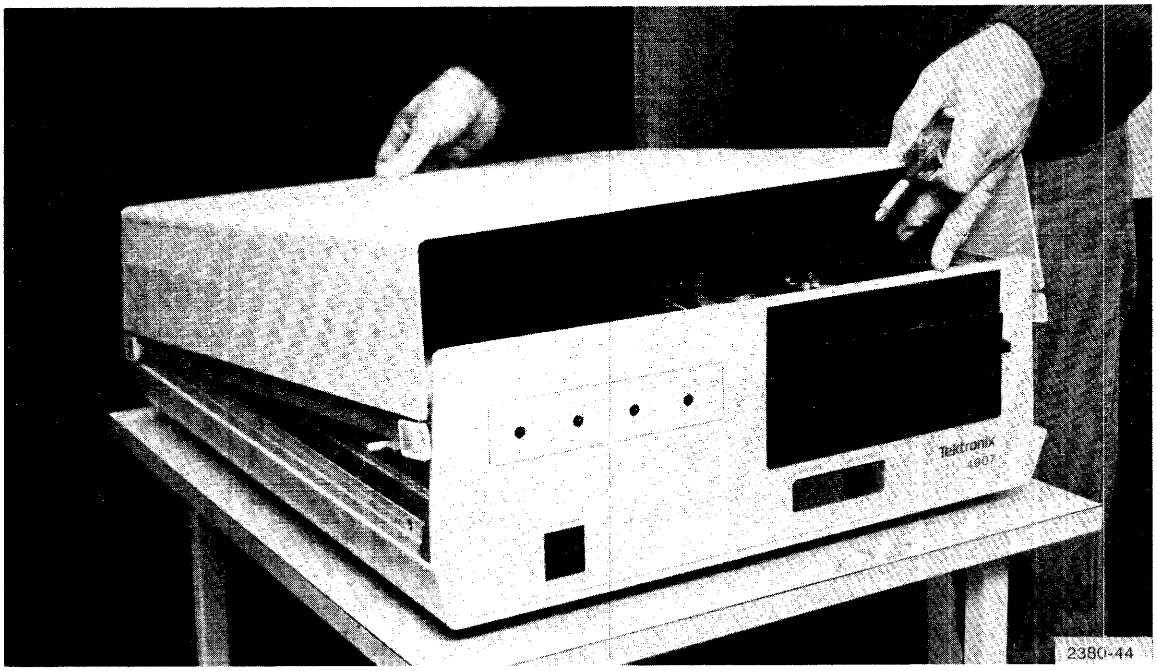


Figure 6-1. Removing 4907 Main Cabinet Cover.

b. Remove oxide from head with swab and alcohol (Figure 6-2).

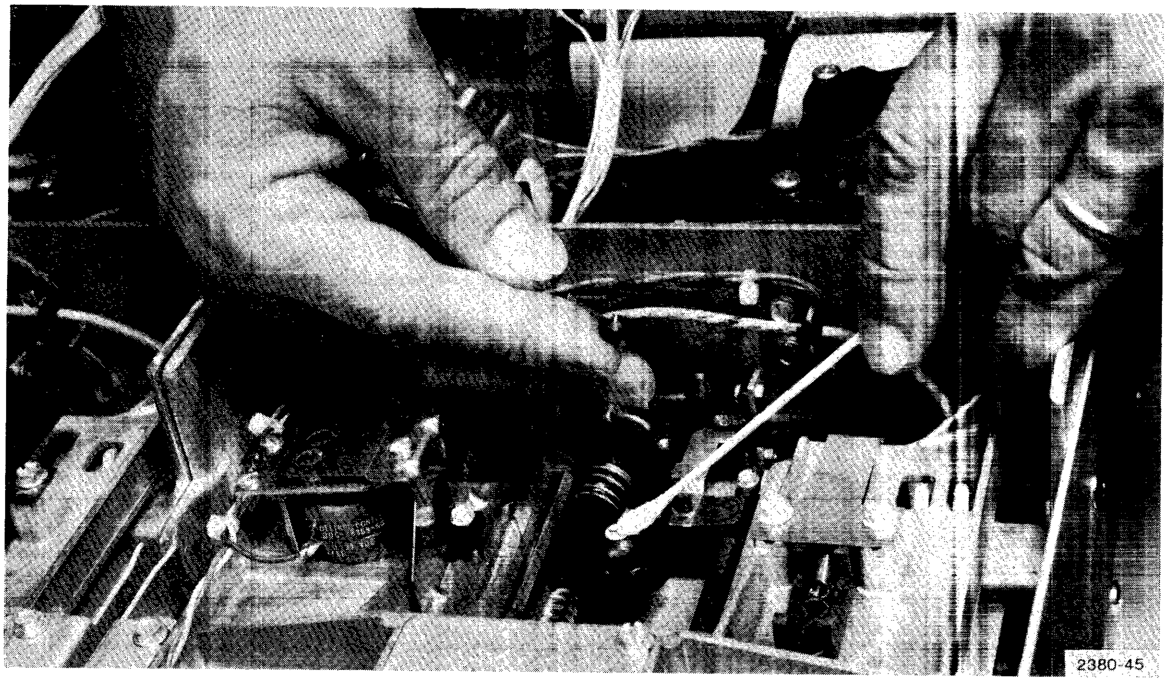


Figure 6-2. Cleaning Read/Write Head.

c. Replace drive cover.

Procedure No. 2

a. Remove cover from drive (Figure 6-3).

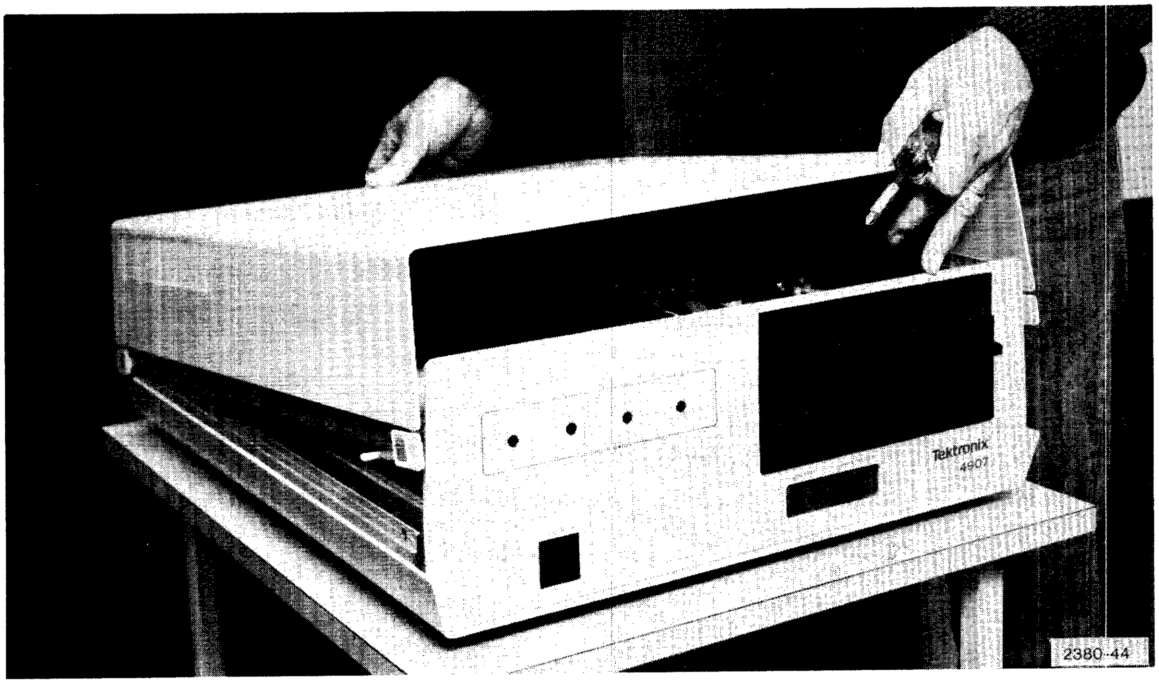


Figure 6-3. Removing 4907 Main Cabinet Cover.

b. Remove read/write head button (Figure 6-4).

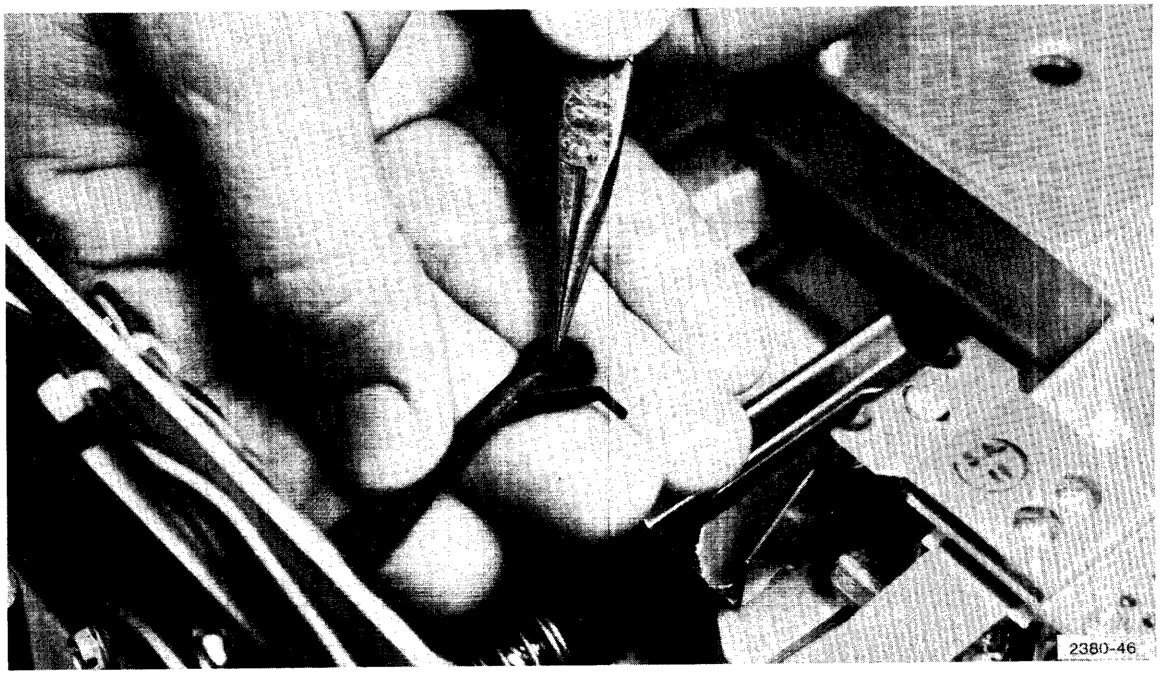


Figure 6-4. Removing Read/Write Head Button.

c. Install new read/write head button (Figure 6-5).

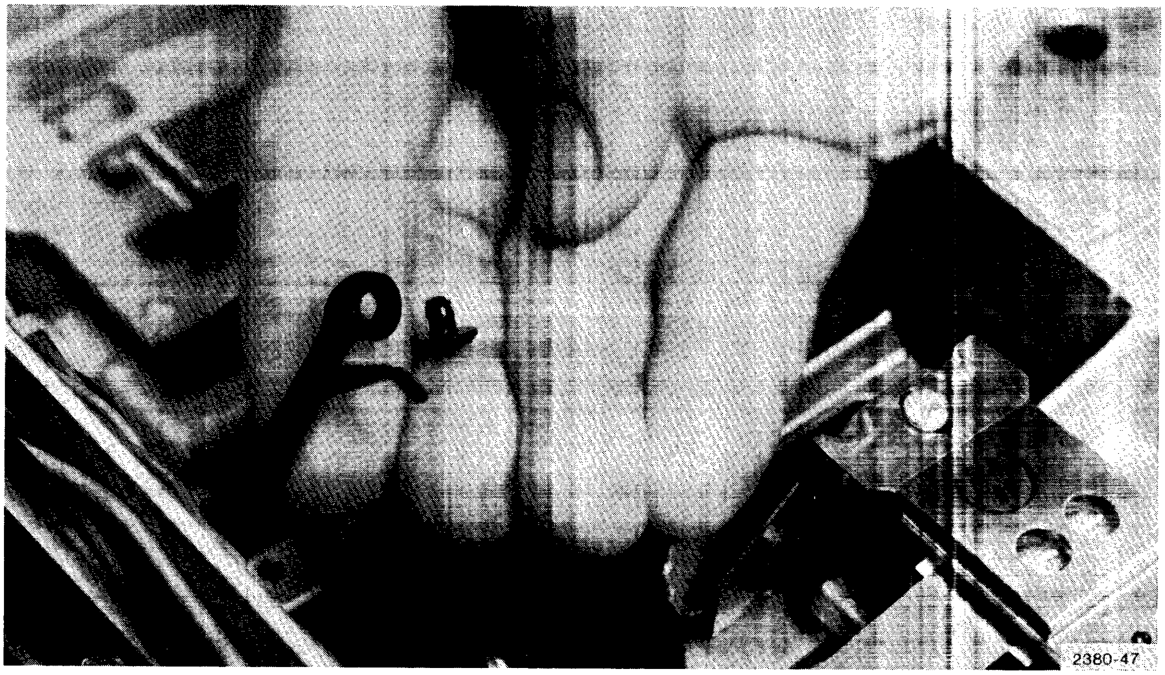


Figure 6-5. Installing New Read/Write Head Button.

d. Replace drive cover.

SECTION 7

CONTENTS

Section 7 **SPECIFICATIONS**

Section 7

SPECIFICATIONS

4907 PERFORMANCE SPECIFICATIONS

1. Data File Storage Capacity (formatted and accessible by operator)

Per Drive (includes 256-byte directory)	630,528 bytes
Per Track	8192 bytes
Per Sector	256 bytes

2. GPIB Data Transfer Rate

Burst	3900 bytes/sec
Sustained	1300 bytes/sec

3. Error Rate

Refer to FLEXIBLE DISC DRIVE SPECIFICATIONS.

4907 PHYSICAL SPECIFICATIONS

1. Dimensions

Height	7.94 in	(20.17 cm)
Width	20.31 in	(51.59 cm)
Depth (minus drive door handle projection)	25.25 in	(64.14 cm)

2. Weight

Main Unit	51 lbs (23.1 kg)
Auxiliary Unit of Option 30	50 lbs (22.6 kg)
Auxiliary Unit of Option 31	62 lbs (28.1 kg)

4907 ENVIRONMENTAL SPECIFICATIONS

1. Operating Environment

Ambient Temperature	50 degrees F to 100 degrees F (10° C to 38° C)
Relative Humidity	20% to 80%
Maximum Wet Bulb	78° (25° C)
Maximum Altitude Above Sea Level	10,000 ft (3048 m)

2. Storage Environment

Ambient Temperature	50° F to 125° F (10° C to 52° C)
Relative Humidity	8% to 80%
Maximum Altitude Above Sea Level	50,000 ft (15,240 m)
Shipping Temperature	−40° F to 125° F (−40° C to 52° C)

3. Shock (nonoperating)

Unit will not suffer damage or fail to operate when subjected to three impact shocks of approximately 20 g's in each direction along each main axis. Shock time is 11 ± 1 ms.

4. Vibration

Unit will not suffer damage or fail to operate when subjected to the following vibration for a period of 5 minutes along each main axis.

Nonoperating	5 to 25 Hz at 0.008 in displacement
	25 to 55 Hz at 0.004 in displacement
Operating	5 to 25 Hz at 0.0014 in displacement
	25 to 55 Hz at 0.0007 in displacement

4907 ELECTRICAL SPECIFICATIONS

1. AC Power Supply Requirements

   A rear panel line voltage selector matches the transformer inputs to four different line voltages. 50 Hz systems can be used by changing the pulley and belt in the disc drives. Table 7-1 shows the allowable line voltages:

Table 7-1

LINE VOLTAGES

Line Voltage	Tolerance	Frequency	Fuse
100 Vac	90-110		2 A Med Blow
120 Vac	108-132	50 or 60 Hz	2 A Med Blow
220 Vac	198-242	± .5 Hz	1 A Med Blow
240 Vac	216-264		1 A Med Blow

2. Maximum power dissipation at 120 Vac, 60 Hz, is 140 W.

3. Maximum load current at 120 Vac, 60 Hz, is 1.7 A.

FLEXIBLE DISC DRIVE SPECIFICATIONS

1. Type

   Rackmount Flexible Disc Drive, with hard sector (32), write-protect hole detect, and double-density recording.

2. Performance Specifications

Capacity (unformatted)	
Per Disk	6.4 megabits
Per Track	83.4 kilobits
Transfer Rate	500 kilobits/sec
Latency (average)	83 ms
Access Time	
Track to Track	8 ms
Average	260 ms
Settling Time	8 ms
Head Load Time	35 ms

3. Physical Specifications

Error Rates	
Soft Read Errors	1 per 10E9 bits read
Hard Read Errors	1 per 10E12 bits read
Seek Errors	1 per 10E6 seek operations

4. Environmental Specifications

Same as 4907 ENVIRONMENTAL SPECIFICATIONS.

MEDIA REQUIREMENTS

1. Type	Double-density compatible
2. Storage Environment	
Temperature	40° F to 140° F (5° C to 60° C)
Humidity	8% to 80%
3. Media Lifetime	
Passes per track	3.5 x 10E5
Insertions	>30,000

ROM PACK SPECIFICATIONS

1. Dimensions

Length	4.66 in (11.836 cm)
Width	2.62 in (6.655 cm)
Depth	0.88 in (2.235 cm)

2. Weight	8 oz (227 gm)

3. Environmental Requirements

   Same as 4907 ENVIRONMENTAL SPECIFICATIONS.

4. Power Requirements (from Graphics System)

   +5 Vdc 300 mA

APPENDIX A

CONTENTS

Appendix A

MESSAGES

DEVICE AND FILE STATUS MESSAGES

File Status Messages

File status messages are generated by CALL "FILE", DIRECTORY, and CALL "DSTAT" commands. The messages are stored in a string variable specified in the command. The messages generated by DIRECTORY may be displayed on the screen, printed, or stored on tape.

A complete or "full" file status message has the following format:

```
BRS  N ATR      NNNNNNN   ALLOC      DD-MON-YY  HH:MM ALT
                NNNNNNN   USED       DD-MON-YY  HH:MM USED
     N OPEN      NNNNNN   REC LEN    DD-MON-YY  HH:MM CREATED
FILE IDENTIFIER
```

Field Descriptions

DD-MON-YY HH:MM ALT
DD-MON-YY HH:MM USED
DD-MON-YY HH:MM CRE

These fields show the day of the month, year, hour and minute the file was LAST ALTERED, LAST USED, and CREATED. If the system clock was not set, the fields look like this:

"00-XXX-00 00:00"

The following table shows which fields are updated when the commands listed are executed.

Fields	Last Altered	Last Used	Created
Commands	CREATE COPY TO (new file only) CALL "DUP" (new file only) SAVE (both files) WRITE (updated when file is closed) PRINT(updated when file is closed) CALL "SPACE" CLOSE (only if opened "F" sequential)	CREATE COPY TO (both files) CALL "DUP" (both files) SAVE (both files) CLOSE ASSIGN APPEND CALL "RENAME"	CREATE SAVE (old file only)

BRS N ATR

Denotes the file attributes, either as specified in the CREATE command, or altered by the ASSIGN command. If a C follows the S in the attribute field, it means that although the file was specified "scattered," the file was created contiguously because plenty of room was available.

NNNNNNN ALLOC

Shows the number of bytes allocated to the file.

NNNNNNN USED

Shows the number of bytes actually used for storage.

N OPEN

Shows the number of current OPENs on this particular file. With single host systems this will always be 1.

NNNNN REC LEN

Shows the number of bytes allocated to each record in a random access file. Shows zero (0) for a sequential file.

FILE IDENTIFIER

Shows the name of the file. This field will appear in the first line of the message if a DIRECTORY was executed.

Example

```
MYLIBRY/TOM/PERSONAL          9984 ALLOC    01-DEC-77 12:30 ALT
          BUS N ATR           6000 USED     01-DEC-77 08:10 USED
              1 OPEN            60 REC LEN  01-DEC-77 08:10 CRE-
ATED
```

This message would appear after executing a DIRECTORY command for a file created at 8:10 AM, last altered at 12:30 PM, with all activity taking place Dec. 1, 1977. Currently there is only one OPEN. The file was allocated 9,984 bytes of space, of which 6000 contain information. The records in the file are 60 bytes long. "BUS N ATR" indicates the file contains binary, public and scattered but non-compressible, information. The F.I. indicates the name of the file is "PERSONAL" and is contained under libraries "TOM" and "MYLIBRY."

General Information

The DIRECTORY command allows the user to generate file status messages with three levels of information:

The first level includes the F.I. only.

The second level includes the F.I. and when the file was created, last used and last altered.

The third level includes the F.I., attributes, space specifications, and when the file was created, last used and last altered.

Device Status Messages

The device status messages are generated by the CALL "DSTAT", CALL "CUSTAT", and CALL "MOUNT" commands and are stored in the specified string variable. The message format is shown below:

```
NN UNIT
XXXXXXXXXXXX DEV ID XXXXXXXXXX VOL ID XXXXXXXXXXXXXXXXXXXXXXXX
OWNER NNNNNNNNN FREE NNNNNNNNN SIZE NNNNNNNNN LOST NNNNN BLK SIZE
DD-MMM-YY HH:MM FORMATTED NNNNN FILES OPEN RESERVED WRITE PROTECTED
```

Field Descriptions

NN UNIT
Shows the device address. This field appears only when CALL "CUSTAT" is executed.

XXXXXXXXXXXX DEV ID
Shows the nomenclature of the device.

XXXXXXXXXX VOL ID
Indicates the name of the disc as specified in the CALL "FORMAT" command.

XXXXXXXXXXXXXXXXXXXXXXXX OWNER
Indicates the owner's or user's name as specified in the CALL "FORMAT" command.

NNNNNNNNN FREE
Indicates the number of usable non-file bytes remaining on disc.

NNNNNNNNN SIZE
Shows the number of storage bytes provided by this type of disc.

NNNNNNNNN LOST
Indicates the number of bytes of unusable or damaged space.

NNNNN BLK SIZE
Indicates the size of each principal block in bytes.

DD-MMM-YY HH:MM FORMATTED
Indicates the day of the month, month, year, hour and minute the disc was formatted.

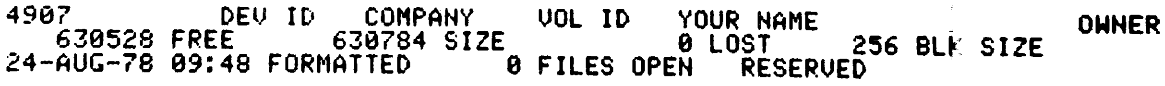
NNNNN FILES OPEN
Shows the number of files currently open.

RESERVED — Appears only if device is reserved.

WRITE PROTECTED — Appears if device or disc is write-protected.

Example

01 UNIT



```
4907          DEV ID    COMPANY    VOL ID   YOUR NAME                  OWNER
   630528 FREE       630784 SIZE           0 LOST      256 BLK SIZE
24-AUG-78 09:48 FORMATTED      0 FILES OPEN   RESERVED
```

This message would be generated for device 1, which is identified as a 4907. The disc is identified as "COMPANY" and the owner as "YOUR NAME." There are 630,528 bytes of usable non-file space. The 630,784-size field indicates this is a flexible disc with no bytes of unusable or damaged space. The block size is 256 bytes.

The disc was last formatted on August 24, 1978 at 9:48 AM. No disc files are open and the device is reserved but not write-protected.

The 01 unit will only appear when CALL "CUSTAT" is executed.

ERROR MESSAGES AND RECOVERY PROCEDURES

(*Asterisks indicate that additional information will be displayed, further defining the error.)

Error Message	Cause	Correction
ERROR 1 BUS I/O ERROR*	GPIB data transfer error often caused by static electricity.	Execute an INIT then reissue command. If this is not successful, restarting program may be necessary.
ERROR 2 ILLEGAL COMMAND	4907 or Graphic System error .	This message may also appear during a WRITE command if a device I/O error has occurred. If so, be sure the 4907 has been loaded and is operating correctly. If error message repeats, see 4907 File Manager Service Manual.
ERROR 3 COMMAND FORMAT	4907 or Graphic System error. See 4907 File Manager Service Manual.	
ERROR 4 ILLEGAL COMMAND FIELDS	One or more entries in the CALL "FORMAT" or CALL "FFRMT" commands are illegal: volume number entry is not 1, number of volumes entry is not 1, number of directory chains is 0 or greater than 10.	Execute command again with correct entries.

Error Message	Cause	Correction
ERROR 5 ILLEGAL MASTER PASSWORD	An attempt has been made to execute a command containing an illegal entry in the master password field.	Master password field can contain no more than 10 characters. The first character must be alphabetic and the rest alphanumeric. No spaces may be entered between characters.
ERROR 7 ILLEGAL FILE IDENTIFIER	An F.I. has incorrect construction or illegal characters.	See FILE IDENTIFIER CONSTRUCTION in Section 4
ERROR 8 ILLEGAL VOLUME IDENTIFIER	The volume I.D. field in a formatting command has illegal characters, incorrectly located spaces, or is blank.	The volume I.D. field can contain no more than 10 characters. The first character must be alphabetic. No spaces may be entered between characters. The field cannot be blank.
ERROR 10 DEVICE NOT FOUND	The system cannot find the device with the address specified in the command.	Re-execute command with correct address. If this message appears the first time the system is used, the adjustable strapping in the controller may have been positioned incorrectly. See 4907 Installation Guide.
ERROR 11 DEVICE WRITE PROTECTED	An attempt was made to write to a disc that is write-protected.	Be sure the device should be written to, and then turn off the write protect switch for the device. Be sure the write-protect tape is covering the write-protect hole on the disc.

Error Message	Cause	Correction
ERROR 12 DEVICE RESERVED	The device was reserved and a command requiring a free device was attempted.	Release the device with a CALL "DREL" command and continue.
ERROR 13 DEVICE NOT RESERVED	The device was not reserved and a command requiring a reserved device was attempted.	Execute a CALL "DRES" command and continue.
ERROR 14 DEVICE HAS FILES OPEN	A CALL "DRES" command was issued to a device with open files.	Close all files on that device and continue.
ERROR 15 DEVICE I/O ERROR	A hard error has occurred on a device.	See CALL "HERRS" command description in Section 5. If a particular location on a disc is causing repeated errors, it may be taken out of use with a CALL "MRKBBG" command. See 4907 File Manager Service Manual for complete details.
ERROR 16 DEVICE NOT READY*	This message appears if: 1. The disc is not in place. 2. The disc has been loaded improperly. 3. The door has not been closed. 4. The device is not up to speed.	Make sure address is correct, that the disc is properly loaded, and that the device is up to speed.

Error Message	Cause	Correction
ERROR 17 DEVICE NOT MOUNTED*	An attempt was made to use an unmounted disc.	Execute a CALL "MOUNT" command and continue
ERROR 18 NO SPACE*	There is not enough space left on the disc for further storage.	Delete any unneeded files (KILL), gather the remaining free space in a single area with the CALL "COMPRS" command.
ERROR 20 CONTROL UNIT ERROR TABLE SPACE EXHAUSTED	A command requiring space equivalent to two files was attempted with either 8 or 9 files already open. This eliminates room necessary for execution of the command.	Close two files and reissue the command.
ERROR 22 CONTROL UNIT PROCESSOR ERROR RAM FAILED	A hardware failure in the controller prevents further system use.	See 4907 File Manager Service Manual.
ERROR 23 CLOCK NOT READY	The system is not operating.	Execute a CALL "SETTIM" command
ERROR 32 DIRECTORY CHAIN DAMAGE*	This may occur when attempting a CALL "MOUNT" command.	Execute CALL "DUP" Then reattempt CALL "MOUNT" on new disc. Information may be unrecoverable on a disc generating this massage.

Error Message	Cause	Correction
ERROR 34 DATA AREA DAMAGE*	This results from damaged or bad blocks on a disc.	Copy the file to another disc or to another area on the same disc. Then: 1. Kill old file. 2. Reserve device (CALL "DRES"). 3. Execute CALL "MRKBBG". 4. Release device (CALL "DREL").
ERROR 35 BLOCK USAGE CONFLICT	The system has incorrectly assigned a physical block to two of these three categories: 1. Directory use. 2. Data use. 3. Bad block group. Since no block can be assigned to more than one use at a time or overlap an adjacent block, this error message appears.	Execute a CALL "DUP" command.
ERROR 40 LOGICAL FILE NUMBER NOT FOUND*	A command execution has been attempted specifying a logical file number that does not exist. This occurs when the lfn in a command does not match the lfn specified in an earilier OPEN command.	Execute an OPEN command with the correct lfn.

Error Message	Cause	Correction
ERROR 41 ACTIVE LOGICAL FILE NUMBER*	An OPEN or OPEN "G" command specifying an illegal lfn has been issued. This lfn has already been assigned to a file which is currently active.	Execute another OPEN command with an unused lfn, or close old file.
ERROR 42 FILE NOT FOUND*	The device involved does not contain a file with the specified name.	Be sure the device address and the F.I. are correct and reissue the command.
ERROR 43 DUPLICATE FILE IDENTIFIER*	An attempt was made to create a new file with an F.I. belonging to an existing file.	Use a different F.I. in the CREATE command.
ERROR 44 LIBRARY NAME CONFLICT	A file cannot be created at any level where there is already a library of the same name.	Change the name of the file and reissue the CREATE command.
ERROR 45 FILE LOCKED*	An attempt was made to open a file without using the passwords specified in the CREATE command.	Reissue the OPEN command using the correct passwords.
ERROR 50 FILE IS WRITE PROTECTED*	An attempt was made to WRITE to a file opened for read access only.	If you do wish to write to the file, close it and then reopen it for "full" access.

Error Message	Cause	Correction
ERROR 51 FILE IS RESERVED*	A GPIB OPEN command was issued without setting the "wait if file reserved" bit.	This message appears when faulty commands not involving the ROM pack have been issued. See 4907 File Manager Service Manual.
ERROR 52 ILLEGAL FILE OPERATION*	An attempt was made to perform an illegal file operation.	Review commands involved in carrying out the operation to be sure they are valid for the file or information being accessed.
ERROR 53 END OF FILE*	An attempt has been made to read past the end of a file.	This message does not appear if an ON EOF command is used.
ERROR 54 ILLEGAL FILE EXPANSION*	An attempt has been made to write additional data to a noncontiguous location on the disc for a file specified "contiguous."	The attribute C (contiguous) may be changed to S (scattered) using the ASSIGN command.
ERROR 55 NO SPACE*	There is not enough uncommitted space left on the disc for further storage.	Delete any unneeded files (KILL), and gather the remaining free space in a single area with the CALL "COMPRS" command. I/O command may then be reissued.
ERROR 56 POINTER NOT AT ITEM BOUNDARY*	This message results from improper system operation.	See 4907 File Manager-Service Manual.

Error Message	Cause	Correction
ERROR 57 ILLEGAL ITEM HEADER*	The TYPE function has encountered an illegal item header for a standard item file.	See TYP command description in Section 5.
ERROR 60 FORMAT FAILED*	If the CALL "FORMAT" command was issued, it has failed because the disc has too many bad blocks or the first block containing the volume label is bad. If the CALL "FFRMT" command was issued, it may have failed because the old volume label was illegible due to damage.	If message appears after attempting a complete, accurate "full" format, choose another disc.
ERROR 61 MARK BAD BLOCK GROUP FAILED*	The CALL "MRKBBG" command failed because the bad block table in the volume label is too full or hardware errors are involved.	Copy files to a new disc or refer to 4907 File Manager Service Manual.
ERROR 62 ILLEGAL ATTRIBUTE CHANGE*	The ASSIGN command contains attribute entries which conflict with the existing file type.	Reissue the ASSIGN command with correct attribute entries.

Error Message	Cause	Correction
ERROR 63 ILLEGAL IPL FILE*	The file name extension is not valid or the file is not a host binary type.	Reissue the command with the correct extension construction.
ERROR 70 COPY SKIPS LOCKED FILE*	A COPY . . . TO command without correct passwords was issued to a locked file.	Reissue the COPY . . . TO command using the correct passwords. If the F.I. uses special characters to copy multiple files, copying will continue with the next file.
ERROR 71 COPY SKIPS OPEN FILE*	A COPY . . . TO command was attempted on an open file.	If you wish to copy this file to another location, it must first be closed. If the F.I. in the command uses special characters to copy multiple files, copying will continue with the next file.
ERROR 74 DELETE SKIPS LOCKED FILE*	A KILL command was issued to a locked file.	Reissue the KILL command. If you wish to delete files with passwords, those passwords must be entered in the F.I. or the master password must be used. If the specific passwords or the master pasword is not used, the locked files cannot be deleted.
ERROR 75 DELETE SKIPS OPEN FILE*	A KILL command was attempted on an open file.	If you do intend to delete this file, close it and then reissue the KILL command.

Error Message	Cause	Correction
ERROR 78 RENAME SKIPS LOCKED FILE*	A CALL "RENAME" command without correct passwords was attempted on a locked file.	Reissue the CALL "RENAME" command with the correct passwords.

COMMAND EXAMPLES

Command	Example	Result
APPEND	APP"@ MYLIBRY/ A":1150,15	Takes the entire program in file "A" in "MYLIBRY" and places it in memory starting with line 1150 of the current program. Line numbers are to be in increments of 15. The "@" suppresses the current library allowing the system to access MYLIBRY.
ASSIGN	ASS"@ YOURLIBRY/ MATH/STAT";"R"	Changes the "STAT" file under libraries YOURLIBRY/MATH to a private status.
CALL "COMPRS" (Compress)	CAL"COMPRS",1,1	Collects space in files on device 1 not containing information as well as non-file space and groups it into larger contiguous blocks.
CLOSE	CLO	Closes all files.
COPY . . . TO	COP"@ YOURLIB/ FRED/TOM",1 TO "@ MYLIBRY/FRED/ TOM",2	Copies file "TOM" from device 1 to device 2. If "MYLIBRY" and "FRED" do not exist on device 2, they are automatically created.
CREATE	CRE"@MYLIBRY/ DATA","A";100,70	Creates the file "DATA" on the library "MYLIBRY." The information is to be stored in ASCII in 100 records of 70 bytes each.

Command	Example	Result
CALL "CUSTAT" (Controller Unit Status)	CAL"CUSTAT",A$	Generates a status message for all devices interfaced to the controller and stores the message in the specified string variable.
DIRECTORY	DIR2,"@ MYLIBRY/ GRADES"	Locates the file "GRADES" in library "MYLIBRY", generates a "full" file status message, and displays it on the Graphic System screen. 2 indicates the format code.
CALL "DISMOUNT"	CAL"DISMOUNT",1	Deactivates device 1. Prevents system use of device until another CALL "MOUNT" is executed.
CALL "DREL" (Device Release)	CAL"DREL",2	Releases exclusive control of device 2.
CALL "DRES" (Device Reserve)	CAL"DRES",2	Reserves device 2.
CALL "DSTAT" (Device Status)	CAL"DSTAT",2,A$	Generates a device status message about device 2, as well as a status message on all open files. This message is stored in A$.
CALL "DUP" (Duplicate)	CAL"DUP",1,2,1	Duplicates all information from device 1 to device 2. All space, including unused file space, is collected into a contiguous space.
CALL "FFRMT" (Fast Format)	CAL"FFRMT",1, "LAB",1,1, "GENETICS", "BLUE",7,7,3,3,3	Operates same as CALL "FORMAT" except no surface analysis is performed. This command is mainly for reformatting discs quickly.
CALL "FILE"	CAL"FILE",2 " MYLIBRY/MATH", A$	Generates file status message about file "MATH" in library "MYLIBRY" on device 2. Message is stored in A$.

Command	Example	Result
CALL "FMVALS" (File Manager Values)	CAL"FMVALS", A,A$	Sends the current device address to A and the name of the current library to A$.
CALL "FORMAT"	CAL"FORMAT",1, "LAB",1,1, "GENETICS", "BLUE",7,7,3,3,3	Formats a disc and creates a volume label with the following information: device address: 1 volume identification: LAB volume number: 1 number in series: 1 owner's name or ID: GENETICS master password: BLUE 1st through 5th level chains: 7,7,3,3,3 Also conducts surface analysis and locates and records unusable space on the disc. This command is mandatory on new discs.
CALL "HERRS" (Hard Error Status)	CAL"HERRS", 1,A,B,C,D	Requests a count of the last I/O retries, the total I/O retries since last power up, number of successful recoveries, and number of unsuccessful recoveries. These all refer to device 1. All numbers generated are stored in variables A, B, C and D, respectively.
INPUT	INP#3,40:A$	Reads the ASCII string in record 40 in lfn 3 and stores it in A$.
KILL	KIL"@MYLIBRY#"	Deletes all closed files in "MYLIBRY."
CALL "MOUNT"	CAL"MOUNT",1,A$	Activates device 1 for system use and generates device status message. Message is stored in A$.

Command	Example	Result
CALL "MRKBBG" (Mark Bad Block Group)	CAL"MRKBBG",2, "COLL","SECRET", "010001FA"	This command is executed only after a message specifying defective disc areas appears. The command identifies the volume I.D. as "COLL" with a master password of "SECRET" on device 2. The address of the defective area is "010001FA."
CALL "NEXT"	CAL"NEXT",3,A$	Closes the current file and opens the next file in a series specified by an OPEN "G" (group) command. This command assigns the lfn 3 to each file (in the group of files) as it is opened. CALL "NEXT" generates a new file status message with each new file, which is stored in A$. If A$ is returned "" as a null string, then no files remain in group.
OLD	OLD"$FINREC/ DEPTB"	Locates the program file "DEPTB" in "SYSLIB/FINREC" and reads it from disc to Graphic System memory.
ON EOF	ON EOF(2)THEN 165	When end of file is encountered in the file represented by lfn 2, system executes line 165.
OPEN	OPE"@UNIV/MAINT-REC";1,"U", A$	Opens the file "MAINTREC" in library "UNIV" for updating. It also assigns 1 as the logical file number which associates this F.I. with the current device. This way when lfn 1 is specified in subsequent commands, the system knows which file on which disc to access.
PRINT	PRI#3,40:A$	Prints the ASCII string in A$ to record 40 in lfn 3.
READ	REA#3,40:A$	Reads binary data in record 40 in lfn 3 and stores it in A$.

Command	Example	Result
CALL "RENAME"	CAL"RENAME",1, " PERSON-NEL/MGRS/ JONES", " PERSON-NEL/MGRS/ SMITH"	Renames the file "JONES" under libraries "PERSONNEL/MGRS" on device 1 to "SMITH."
CALL "REWIND"	CAL"REWIND",1	Rewinds pointer on the file associated with lfn 1 in an earlier OPEN command to beginning of file.
SAVE	SAV"@B10/ CHEMSTAT"; 100,-3150	Transfers a current program to the file "CHEMSTAT" in library "B10" starting with line 100 and ending with line 3150 in binary format. If no file exists, the command automatically creates it.
SECRET	SEC	Prevents future program listing.
CALL "SETTIM" (SET TIME)	CAL"SETTIM", "4-JUL-78 16:30:20"	Sets system clock to 20 seconds after four-thirty PM on July 4, 1978.
CALL "SPACE"	CAL"SPACE", 3, 3000, A,B	Adjusts file space to 3000 bytes on the file identified as lfn 3 in an earlier OPEN command. Also generates two messages: the actual number of bytes required to store data and the actual number of bytes allocated. These messages are stored in A and B, respectively.
CALL "TIME"	CAL"TIME",A$	Sends the current system time to A$.
TYP (function)	T=TYP(2)	Performs type function on file associated with lfn 2. See TYP command description in the 4050 Series Graphic System Reference Manual.

Command	Example	Result
UNIT	UNI 3	Specifies device 3 as the current device.
CALL "UNIT"	CAL"UNI",B	Specifies the device address in B as the current device.
CALL "USERLIB" (User Library)	CAL"USERLIB", "PORT-LAND/INDUSTRY"	Causes "PORTLAND/INDUSTRY" to be the current library. Unless this current library is circumvented with special characters, the system will attempt to locate all files in "PORT-LAND/INDUSTRY."
WRITE	WRI #3,40:A$	Writes the binary string in A$ to record 40 in lfn 3.

APPENDIX B

Appendix B SAMPLE PROGRAMS

Appendix B

SAMPLE PROGRAMS

These programs are useful in general 4907 File Manager operation. They also demonstrate typical use of many system commands.

1. RANDOM FILE INITIALIZING PROGRAM

This is a two-part program. The first part prompts the user to enter start-up sequence information required on a day-to-day basis. This information is automatically entered in the commands shown in the program listing below. The program will

110	Initialize the Graphic System.
140	Designate the current device.
150	Mount the device.
190	Designate the current library (CALL "USERLIB").
210	Specify the F.I.
270	Kill (delete) the file if desired.
280-400	Create a random file and establish the number of records and record length. See CREATE command description for details on determining record length.
280-400	Create a sequential file if desired.
430	Open the file.
420-550	Enter blank strings in all records or random files. See Step 9 of GENERAL SEQUENCE FLOW CHART at the end of this manual.

```
100 PRINT "IRANDOM FILE INITIALIZATION PROGRAM"
110 INIT
120 PRINT "JENTER DEVICE ADDRESS:";
130 INPUT A
140 CALL "UNIT",A
150 CALL "MOUNT",A,A$
160 PRINT "ENTER LIBRARY OR LIBRARIES <DOG/CAT>:";
170 INPUT A$
180 IF A$="" THEN 200
190 CALL "USERLIB",A$
200 PRINT "ENTER NAME OF FILE:";
210 INPUT A$
220 CALL "FILE",A,A$,B$
230 IF B$="" THEN 280
240 PRINT "IF FILE EXISTS DO YOU WANT IT DELETED? <Y OR N>:";
250 INPUT B$
260 IF B$<>"Y" THEN 200
270 KILL A$
280 PRINT "ENTER RECORD LENGTH:";
290 INPUT A
```

```
300 PRINT "ENTER NUMBER OF RECORDS:";
310 INPUT B
320 PRINT "IS THIS AN ASCII FILE? (Y OR N):";
330 C$="A"
340 INPUT B$
350 IF B$="Y" THEN 370
360 C$=""
370 PRINT "OTHER ATTRIBUTES THAN ";C$;"RSN? ENTER CHARACTERS, IF ANY:"
380 INPUT B$
390 B$=B$&C$
400 CREATE A$,B$;B,A
410 IF A=0 THEN 570
420 OPEN A$;1,"F",B$
430 DIM D$(A-1)
440 D$=""
450 FOR I=1 TO A-1-4*(C$<>"A")
460 D$=D$&""
470 NEXT I
480 IF C$="A" THEN 530
490 FOR I=1 TO B
500 WRITE #1,I:D$
510 NEXT I
520 GO TO 560
530 FOR I=1 TO B
540 PRINT #1,I:D$
550 NEXT I
560 CLOSE 1
570 END
```

2. ASCII DATA TRANSFER FROM TAPE TO DISC FILE

This program prints ASCII data from an internal Graphic System tape to a specified sequential disc file. A description of significant lines follows the listing.

```
100 INIT
110 PRINT "LGGGITAPE TO DISC PROGRAM FOR DEVICE 0"
120 UNIT 0
130 REM IS DATE SET OR DISC MOUNTED?
140 CALL "TIME",A$
150 IF A$<>"" THEN 190
160 PRINT "DATE AND TIME:";
170 INPUT A$
180 CALL "SETTIM",A$
190 PRINT "JJINSERT DISC IN 4907 AND CLOSE DOORGG"
200 PRINT "IPRESS 'RETURN' WHEN READY";
210 INPUT A$
220 CALL "MOUNT",0,B$
230 PRINT "JJENTER LIBRARY OR LIBRARIES (DOG/CAT):";
240 INPUT A$
250 CALL "USERLIB",A$
260 PRINT "ENTER SOURCE FILE NUMBER FROM TAPE:";
270 INPUT N
280 FIND N
```

```
290 T=TYP(0)
300 PRINT "ENTER NAME OF DISC FILE ONLY:";
310 INPUT A$
320 CALL "FILE",0,A$,B$
330 IF B$="" THEN 390
340 PRINT "SHOULD EXISTING FILE DATA BE ERASED?(Y OR N):";
350 INPUT B$
360 IF B$="N" THEN 440
370 OPEN A$;1,"F",B$
380 GO TO 450
390 T=TYP(0)
400 IF T<2 THEN 230
410 T$=SEG("ABB",T-1,1)
420 CREATE A$,T$;256,0
430 REM FILE SIZE WILL BE INCREASED AS NECESSARY
440 OPEN A$;1,"U",B$
450 ON EOF (0) THEN 600
460 GO TO T OF 490,510,540,570
470 PRINT "EMPTY FILE"
480 GO TO 610
490 PRINT "END OF FILE"
500 GO TO 610
510 INPUT @33:A$
520 PRINT #1:A$
530 GO TO 510
540 READ @33:A
550 WRITE #1:A
560 GO TO 540
570 READ @33:A$
580 WRITE #1:A$
590 GO TO 570
600 PRINT "J̲J̲DONE!G̲G̲G̲"
610 END
```

100 INIT

Returns all variables in the Graphic System to an undefined state.

120 UNIT 0

Tells the system that device 0 will contain the disc and will be the default or current device.

140 CALL "TIME", A$

Requests the time indicated by the system clock and stores it in A$

150 IF A$ < >"" THEN 190

If A$ contains nothing or a null string, then 160 is executed. If A$ contains a time message, then 190 is executed.

160 PRINT "DATE & TIME:";

Requests date and time; i.e., 08-12-78 10:30:30.

170 INPUT A$

System places the date and time entry in A$.

180 CALL "SETTIM",A$

Sets system clock.

Now place the flexible disc in the 4907, close the door, and press RETURN.

220 CALL "MOUNT",0,B$

Tells the system that device 0 is ready for use. A device status message is sent to B$.

230 PRINT "JIENTER LIBRARY OR LIBRARIES (DOG/CAT):";

Requests the name of the library or libraries. If your file is directly under the first level file SCRATCHLIB, press RETURN.

240 INPUT A$

This places the library name (or the null string resulting from pressing RETURN) in A$.

250 CALL "USERLIB", A$

The library name in A$ is specified as the current default library.

260 PRINT "ENTER SOURCE FILE NUMBER ON TAPE:";

Requests the number of the file on the magnetic tape.

```
270 INPUT N
```

Places the number of the file in N.

```
280 FIND N
```

Locates the magnetic tape file specified in N.

```
290 T=TYP(0)
```

Checks the type of information in the tape file.

```
300 PRINT “ENTER NAME OF DISC FILE ONLY:”;
```

Requests the name of the file.

```
310 INPUT A$
```

Places the file name in A$.

```
320 CALL “FILE”, 0, A$, B$
```

Selects the file named in A$ on the current device 0 and sends a file status message to B$. B$ must be dimensioned large enough to contain the status message. (DIMB$(200)).

```
330 IF B$= “” THEN 390
```

If B$ contains no message then 390 is executed.

```
340 PRINT “SHOULD EXISTING DATA BE ERASED? (Y OR N):”;
```

Requests status of current file information.

```
350 INPUT B$
```

Y or N is placed in B$.

```
360 IF B$ =“N” THEN 440
```

If current file information is not to be destroyed, the file is opened with an update (U) access in line 440.

```
370 OPEN A$; 1, "F", B$
```

The file specified in A$ is opened for full access (F) and assigned the logical file number 1.

```
400 IF T<2 THEN 230
```

If the TYP function in 390 returns an integer indicating a file not open or empty or EOF , then a new library name and/or file name must be entered.

```
420 CREATE A$, T$; 256, 0
```

A new file is created at the location of the old disc file. Its name is the same as the name specified in 370. It is specified binary or ASCII, depending on the SEG command in 410. Line 420 specifies this file as a 256 byte sequential file.

```
440 OPEN A$; 1, "U", B$
```

Opens the file created in 420 for update access only.

```
450 ON EOF (0) THEN 600
```

Sets the flag so the progarm will end when end-of-file is reached.

```
460 GO TO T OF 490, 510, 540, 570
```

If end of file is encountered 490 then 610 are executed.

If file contains numeric or character string data (ASCII) 510 is executed.

If file contains numeric data (Binary) 540 is executed.

If file contains Binary string data 570 is executed.

```
480 GO TO 610
```

If TYP encounters an empty file then 470 and 480 are executed.

```
490 PRINT "END OF FILE"

500 GO TO 610

510 INPUT @33:A$
```

520 PRINT#1: A$

530 GO TO 510

The commands 510 thru 530 input the data from the tape file, one record at a time, to the Graphic System, print the data from the Graphic System to the disc file, then repeat through each subsequent record until EOF is encountered.

Lines 540 thru 590 accomplish the same input/output functions as lines 510 thru 530, but for different formats as described under line 460.

610 END

Closes all files and returns control to the Graphic System.

APPENDIX C

CONTENTS

Appendix C **SAMPLE I/O PROCEDURES**

Appendix C

SAMPLE I/O PROCEDURES

USING PRINT AND INPUT IN A PROGRAM

The following programs show how the PRINT and INPUT statements can be used to enter and then retrieve entire strings or numeric values within the strings. It is assumed that the files already have been created and opened.

Retrieving Entire Strings From a Sequential File

```
100    PRINT #3:"BLACK, TOM, 241 MAIN, MILL CITY, 5464451"
110    PRINT #3:"BLUE, IRIS, 566 ELM, TWITVILLE, 6475579"
120    CALL "REWIND",3
130    INPUT #3:A$,B$
140    PRINT A$,B$
```

This program prints these strings on the screen:

```
        BLACK, TOM, 241 MAIN, MILL CITY, 5464451

          BLUE, IRIS, 566 ELM, TWITVILLE, 6475579
```

Line 100 Stores all data shown (as a single string at the beginning of logical file number 3).

Line 110 Stores the data shown (as a single string following the previous string).

Line 120 Returns the logical file pointer to the beginning of the file.

Line 130 Reads the first string from logical file number 3 and stores it in A$; then reads the second string and stores it in B$.

Line 140 Prints the contents of A$ and B$ to the screen.

Retrieving Numeric Values From a Sequential File

```
100    PRINT #3: "BLACK, 200000"
110    PRINT #3: "RED, 600"
120    CALL "REWIND", 3
130    INPUT #3: A, B
140    PRINT A, B
```

This program prints these values to the screen:

```
200000      600
```

Line 100 Stores all data shown (as a single string at the beginning of logical file number 3).

Line 110 Stores all data shown (as a single string following the first string).

Line 120 Sets the logical file pointer back to the beginning of the file.

Line 130 Looks for the first numeric value in the file and stores it in A; then looks for the second numeric value and stores it in B.

Line 140 Prints the values of A and B to the screen.

Retrieving Numeric Values From a Random File

```
100    PRINT #3, 1: "BLUE, 150"
110    PRINT #3, 2: "GREEN, 200"
120    INPUT #3, 1: A
130    INPUT #3, 2: B
140    PRINT A, B
```

This program prints these values to the screen:

```
150    200
```

Line 100 — Stores all data shown (as a single string at the beginning of record 1 in logical file number 3).

Line 110 — Stores all data shown (as a single string at the beginning of record 2 in logical file number 3).

Line 120 — Searches through the string in record 1, locates the first numeric value, and stores it in A.

Line 130 — Searches through the string in record 2, locates the first numeric value, and stores it in B.

Line 140 — Prints the values of A and B to the screen.

NOTE

Printing to a random ASCII file must be done carefully. If more data is transmitted than can be accommodated in the specified record, the data in the following record will be overwritten.

USING WRITE AND READ IN A PROGRAM

The following programs illustrate how standard items can be written to a file, read out, assigned to target variables, and printed to the screen. It is assumed the files have been previously created and opened.

Retrieving Both Strings and Numbers From a Sequential File

```
100    WRITE #3:"RED", "BLUE",145
110    CALL "REWIND",3
120    READ #3:A$,B$,A
130    PRINT A$,B$,A
```

The program prints these strings and this numeric value to the screen:

```
RED      BLUE      145
```

Line 100 — Stores the three data items at the beginning of logical file number 3.

Line 110 — Returns the logical file pointer to the beginning of the file.

Line 120 — Reads the first item and stores it in A$, reads the second item and stores it in B$ and reads the numeric value and stores it in A. If the target variables are out of order (for example, A, A$, B$, or A$, A, B$), an error message appears.

Line 130 — Prints contents of A$, B$, A to the screen in that order.

Retrieving Only Numbers From a Sequential File

```
100   WRITE #3: "WHITE", 123, "RED", 456, "YELLOW", 789
110   CALL "REWIND", 3
120   READ #3: A$, A, A$, B, A$, C
130   PRINT A, B, C
```

The program prints the numeric values only:

```
123      456      789
```

Line 100 Stores the six data items at the beginning of logical file number 3.

Line 110 Returns the logical file pointer to the beginning of the file.

Line 120 Reads each data item in turn and stores it in the specified target variable. "WHITE" goes to A$, 123 goes to A, and so on. The order of the target variables must match the order of the data strings, or an error message will appear.

Line 130 Prints only the values of A, B, and C to the screen.

FILE POINTER OPERATION

Since pointer locations may be altered only in sequential files, this discussion is limited to that type of file. For clarification, however, a table showing random file pointer locations is included.

Each file is equipped with a "pointer." Although this pointer is invisible to the user, it tells the Graphic System where an I/O operation such as a READ or WRITE is to begin. When a file is created, the pointer is automatically positioned at the beginning of the file. Where the pointer is positioned after that depends on the access method ("F", "R" or "U") specified in the OPEN command. For example, if data is being entered into a new file for the first time, an "F" must be specified in the OPEN statement. The pointer will start at the beginning of the file as shown in Figure C-1.

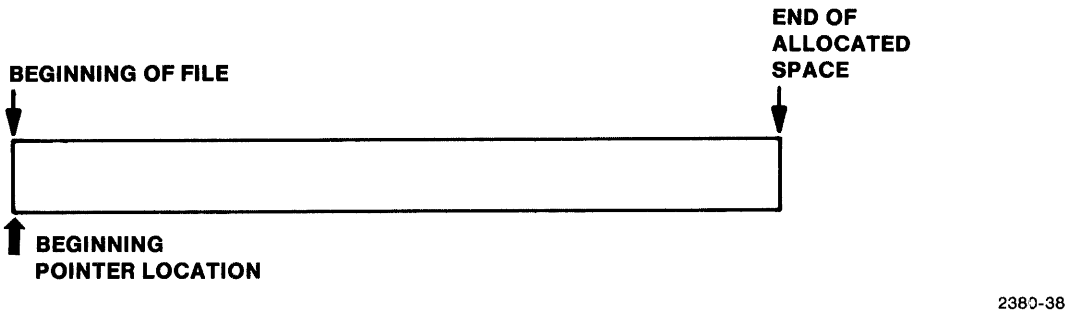


Figure C-1. Graphic Representation of Files.

Once data has been entered, the pointer remains where it was when data entry ended (Figure C-2).

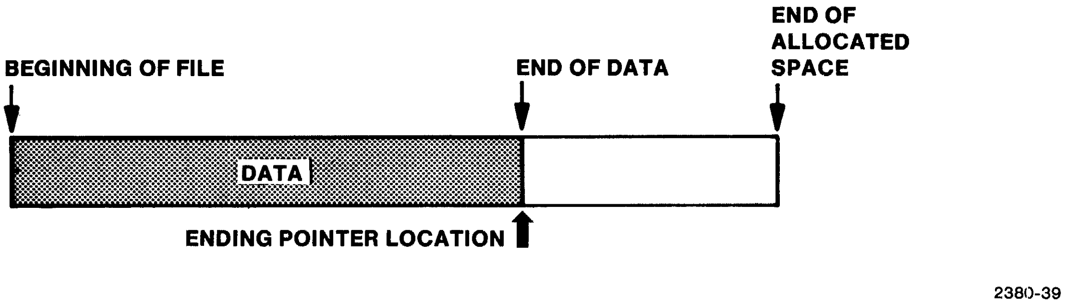


Figure C-2. Pointer Location at the End of Data Entry After an OPEN "F".

The next operation may be started with an OPEN, with an "R" (READ), or with a "U" (UPDATE). If the "U" is entered in an OPEN, the pointer remains at the end of the data. As data is entered, the pointer moves along until data entry is complete. The pointer is still at the end of the data, which in Figure C-3 is also the end of the file.

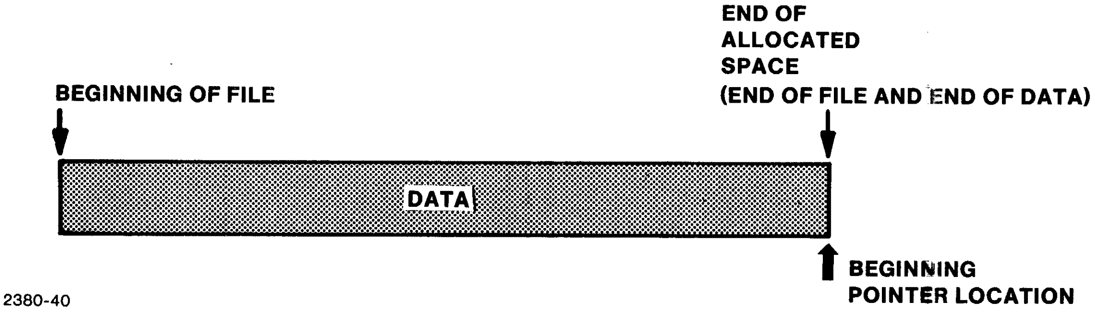


Figure C-3. Pointer location at the end of data entry after an OPEN "U".

If an "R" is entered instead of a "U," the pointer is located at the beginning of the file (Figure C-4).

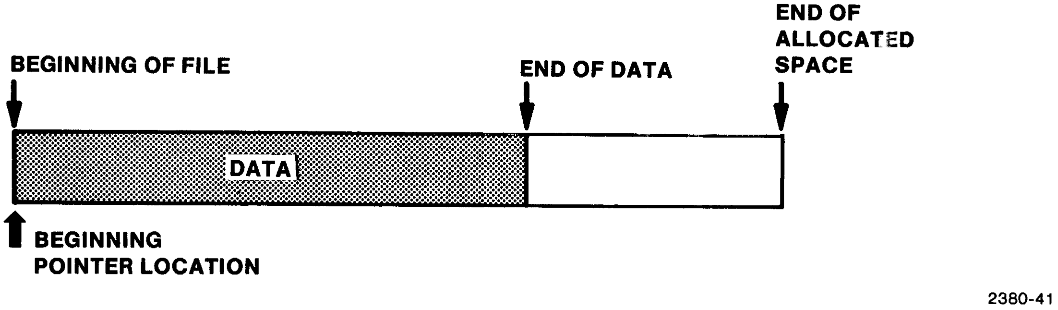


Figure C-4. Pointer location prior to I/O command after an OPEN "R".

READ may now be executed. When all the data has been read, an end-of-file message will be displayed.[1] This message indicates that the pointer has reached the end of the data. If, at this time, you wish to enter more data, you must execute a CLOSE and another OPEN with a "U" for UPDATE.

[1] When an EOF is encountered in a program, the program will abort unless an ON EOF command was executed previously.

Rewinding

Normally, the pointer cannot be moved backward to the beginning of a sequential file unless a CLOSE and a subsequent OPEN are carried out. There are, however, situations when convenient pointer movement is desirable, for example, when you wish to READ or reenter data just entered. A CALL "REWIND" command is provided to reposition the pointer to the beginning of the file; for example, if a user has completed data entry, the pointer would be located as shown in Figure C-5.

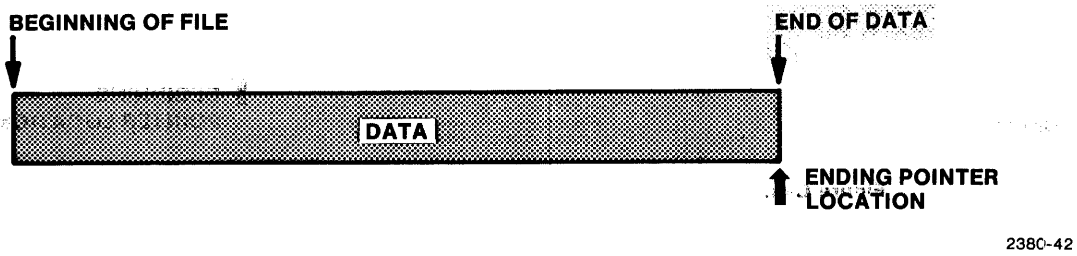


Figure C-5. Pointer Location after Data Entry Before CALL "REWIND".

If it is necessary to READ or reWRITE to the file, the CALL "REWIND" command can be executed. This will move the pointer back to the beginning of the file (Figure C-6).

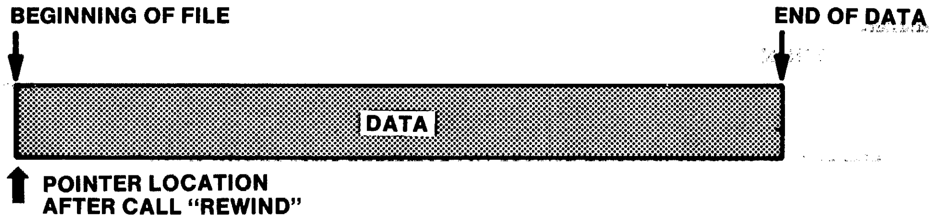


Figure C-6. Pointer Location after CALL "REWIND".

The user may now execute a READ or WRITE without the necessity of a time-consuming CLOSE and OPEN. It is important to remember that if a WRITE is executed after a CALL "REWIND," all data already in the file will be erased.

The following tables show the effects of "F", "R" and "U" entries in OPEN commands for both sequential and random files.

Table C-1

SEQUENTIAL FILE POINTER LOCATION

Character Entered in OPEN Command	Is PRINT or WRITE Allowed?	Initial Pointer Location	Pointer Location When File CLOSED
F (Full)	Yes	Beginning of file	Current position
R (Read)	No	Beginning of file	Current end of data
U (Update)	Yes	End of data	Current end of data

Table C-2

RANDOM FILE POINTER LOCATION

Character Entered in OPEN Command	Is PRINT or WRITE Allowed?	Initial Pointer Location	Pointer Location When File CLOSED
F or U (Full or Update)	YES	Always at beginning of record specified in the I/O command	Current end of data
R (READ)	NO	Always at beginning of record specified in the I/O command	Current end of data

APPENDIX D

CONTENTS

Appendix D

GLOSSARY

ADDRESS (noun)

Refers to number assigned to a particular disc device (device address) through setting of adjustable strapping, or to a variable or to a file as indicated in an F.I. or by a lfn.

ADDRESS (verb)

The act of communicating with a file, device, variable, etc.

ASCII

(American Standard Code for Information Interchange) A standardized code of alphanumeric characters, symbols, and special control characters. Each character is represented by eight binary bits (a byte) in which seven of the bits distinguish one character from another and the eighth bit is reserved for parity checking purposes. The system ignores the parity bit in any ASCII characters it receives, and transmits low parity. (That is, it always transmits a 0 for the eighth bit.)

ASCII FILE

The data in an ASCII file is a sequence of seven-bit ASCII characters. The high order bit is always set to zero. If a number is to be stored, it is converted to a sequence of ASCII characters.

BAD BLOCK GROUP

An area on the disc which, because of damage or surface defects, should not be used for storage. See CALL "FORMAT" and CALL "MRKBBG" command descriptions.

BASIC

An acronym derived from "Beginners All-Purpose Symbolic Instruction Code." BASIC is the high-level programming language used to command a Tektronix 4050 Series Graphic System.

BIT

A binary digit; a 1 or a 0.

BYTE

A group of eight binary bits.

CHAINS

The number of connections between a library at one level and the libraries or files on the following levels. The number of chains dictates the speed at which the system locates files.

CLOSED FILE

See FILE, OPEN.

COMMAND

A line of instructions containing a keyword(s), address(es) and/or an argument. Usually issued from Grapihc System keyboard as contrasted to a statement which is issued from a program.

CONSTANT

Any entry in a field requiring a literal number rather than a representative variable. For example, the entry for the lfn in a command that is not a "CALL" command must be an integer from 1 to 9 and not A, B, etc.

CONTIGUOUS FILE

A file that is not scattered over different locations on the disc but is contained within a single area.

CONTROLLER

The device which, upon receiving instructions from the host, activates peripherals such as drives, plotters, printers, etc., depending on the interface within the controller.

CURRENT DEVICE

The device specified in a UNIT or CALL "UNIT" command. Unless otherwise specified, the default current device is device 0.

CURRENT LIBRARY

The library or libraries specified in a CALL "USER-LIB" command. The default library is "SCRATCH-LIB."

DEFAULT

The value or values which the system may assign to the fields in a command when no entry is made.

DELIMITER

The character(s) or space(s) used to separate one field from another.

DEVICE

A disc drive containing a disc.

DEVICE ADDRESS

The unique number assigned to a particular disc device through setting of adjustable strapping.

DEVICE, RELEASED

A device available for any operation except formatting (see DEVICE, RESERVED).

DEVICE, RESERVED

A device reserved for formatting.

DIMENSIONING

Enlarging or reducing of string variable space to accommodate more or less than the 72 character default length.

DISC

The recording medium used within a drive. Also may be referred to as the "volume", or when residing in the drive, the "device."

DISC, MOUNTED

A disc must be mounted before it can be accessed. The MOUNT command reads the device status message from the disc label, and informs the system that device is available. The DISMOUNT command disables the specified device until another MOUNT command is issued.

DISMOUNT

See DISC, MOUNTED.

DRIVE

The apparatus containing the disc.

ENTRY

The number or character or string placed in a field or the variable or string variable representing it. Entries are made through program operation or keyboard.

ERROR CONDITION

The status of the system when an incorrect entry or command is encountered; generally, when system error conditions occur, error messages are printed to the screen.

EXTENSION

The identifying "tag" placed on the end of file names. Extensions allow the system to distinguish between otherwise identical file names.

F.I.

File Identifier. Refers to the combination of library names and file names required when creating and locating files. F.I.'s are required in many system commands.

FIELD

A portion of a command or statement requiring a specific entry separated from preceding and subsequent fields with delimiters.

FILE

A collection of data, data items or program statements recorded on a storage medium.

FILE IDENTIFIER

See F.I.

FILE, LOGICAL END OF

An EOF mark is automatically placed at the end of all information entered into a file regardless of allocated space. When READ, INPUT or OLD commands are executed, this EOF is encountered and signals the controller that all information has been transmitted. This also may be referred to as END OF DATA.

FILE, OPEN

A data file must be opened before it can be accessed (see OPEN command description in Section 5). When a file is opened, the file pointers are set and a logical file number is assigned. When a file is closed (see CLOSE command), the lfn and file pointers are set to an undefined state. See also FILE POINTER OPERATION in Appendix C.

FILE, PHYSICAL END OF

The end of the space allocated to a file by the CREATE command.

FILE POINTER

Although invisible to the user, the pointer dictates where an I/O operation is to begin. The pointer may be manipulated with an OPEN or CALL "REWIND" command. See FILE POINTER OPERATION (Appendix C).

HEADER

The area preceding each binary data item describing the following item as a string or numeric value. The header also indicates the amount of space used by the item. Standard items are placed into files only with a WRITE command.

HOST

The Graphic System interfaced to the 4907 controls all system activity with the exception of system front panel switches.

HOST BINARY

The Graphic System saves programs in either ASCII or binary form. The binary form is host binary, which means it can be entered only by another Graphic System. ASCII formats usually can be interpreted by all BASIC-based hosts.

KEYWORD

The entry in the first field of all system commands. The keyword tells the system the kind of activity to be carried out.

LFN

Also lfn. The acronym for logical file number.

LIBRARY

A "directory" or "menu" containing the listing and location of subsequent libraries or files. A library itself cannot contain data or programs.

LOCKED FILE

A file that has been assigned a password in its F.I.

LOGICAL FILE NUMBER

The integer from 1 to 9 assigned in the OPEN command that represents both the F.I. and the device address. Up to 9 files may be open at one time. Logical file 0 specifies the internal Graphic System magnetic tape.

MASTER PASSWORD

The master password entered in the CALL "FORMAT" command allows file access in CALL "MRKBBG" and KILL commands without use of specified passwords.

MEMORY

The storage area in the Graphic System used to contain programs.

MOUNT

See DISC, MOUNTED.

NUMERIC EXPRESSION

See 4050 Series Graphic System Reference Manual.

OPEN FILE

See FILE, OPEN.

PASSWORD

The optional entry that may be placed in the F.I., in the CREATE command, immediately following any library or file name at any of the five storage levels. Subsequent access to files is prevented or limited unless the specified passwords are used. See PASSWORDS in Section 4.

POINTER

See FILE POINTER OPERATION (Appendix C).

RANDOM ACCESS

The method of storing or locating data by specifying a record number and directly accessing that record only. Opposite of sequential access.

RANDOM FILE

A type of file with two or more records of identical length.

RECORDS

Two or more divisions of equal length in a random file. Number and length of records are specified in the CREATE command.

SCATTERED

The state of a file not in a contiguous form. File information is stored in scattered locations throughout the disc. Scattered files cannot extend over more than a single disc.

SCRATCHLIB

The system name of the default current library.

SEQUENTIAL ACCESS

The method of storing or locating data by placing the file pointer at the beginning of the designated file and reading or writing towards the end of the file until required information is entered or encountered.

SEQUENTIAL FILE

A type of file with no divisions or "records."

SPECIAL CHARACTERS

Characters that may be used to simplify file access, to provide multiple file selection or in multiple file name changing. See SPECIAL CHARACTERS in Section 4.

STATEMENT

A command preceded by a line number for program use.

STORAGE STRUCTURE

A collection of libraries and files. A storage structure may be as small as a single file.

STRING VARIABLE

See the 4050 Series Graphic System Reference Manual.

SUBFIELD

If two or more entries may be made in a single field, they may be considered "subfields;" for example, the library and file names in an F.I. are subfields.

SYSLIB

The system name of the 1st level system library. Generally used for storing commonly used public programs, data, etc.

SYSTEM

The "system" referred to in this manual includes the disc controller, the disc drive or drives, and the host (Graphic System).

SYSTEM CLOCK

The device in the controller used to record the time and date of various device and file management activities.

TIME/DATE STAMP

The time and date information that is automatically recorded on discs and files when device and file activity commands are executed. The system clock must be operating or no time/date stamps are recorded.

UNIT NUMBER

The number assigned in a UNIT command specifying which device is to be the current device.

USERLIB

The system name of the 1st level user's library. Generally used for any private data or program or that information specific to a particular subject. Unlike SYSLIB or SCRATCHLIB, USERLIB libraries can have any name desired: MYLIBRY, YOURLIBRY, A, DOG, etc.

VARIABLE

See the 4050 Series Graphic System Reference Manual.

VOLUME

Refers to the storage medium used within the drive. Also may be referred to as the "disc."

VOLUME LABEL

Wherever a disc is formatted, a "label" is automatically created, containing information about that disc. See DEVICE AND FILE STATUS MESSAGES (Appendix A).

INDEX

TEKTRONIX®
committed to technical excellence

MANUAL CHANGE INFORMATION

PRODUCT 4907 FILE MANAGER 070-2380-01	CHANGE REFERENCE C1/379 DATE 3-12-79

CHANGE:	DESCRIPTION

EFF ALL SN:

TEXT CORRECTION

PAGE 2-2, GPIB Cable and ROM Pack Installation, after Item 3:

ADD:

NOTE

Whenever GPIB cable is disconnected from 4050 series instrument, be sure to remove ROM Pack as well. Otherwise, when any other instrument asserts SRQ, the 4050 series instrument will hang busy.

1. Set System Clock

The CALL "SETTIM" command starts the system clock so the time and date of various system activities are recorded. It is required each time the system is powered up. There is no "default" time provided.

2. Choose Device

The UNIT command identifies the "current" device. The system accesses the current device whenever it receives a command that does not specifically name some other device. The address to be used in this command can be seen on or around the face of the device. Device 0 will always be accessed if no UNIT command is executed.

3. Has Disc Ever Been Formatted?

All discs must be formatted before they are used for the first time or when all existing information is to be replaced. If the disc already has been formatted and has valid volume label information, no formatting is necessary. CALL "MOUNT" notifies the system that there is a formatted disc ready for use.

4. Reserve Device

You must reserve the device with CALL "DRES" before formatting can take place.

5. Format Disc

The disc may now be formatted with the CALL "FORMAT" command. This command also executes an automatic CALL "MOUNT" to notify the system that there is a formatted disc ready for use.

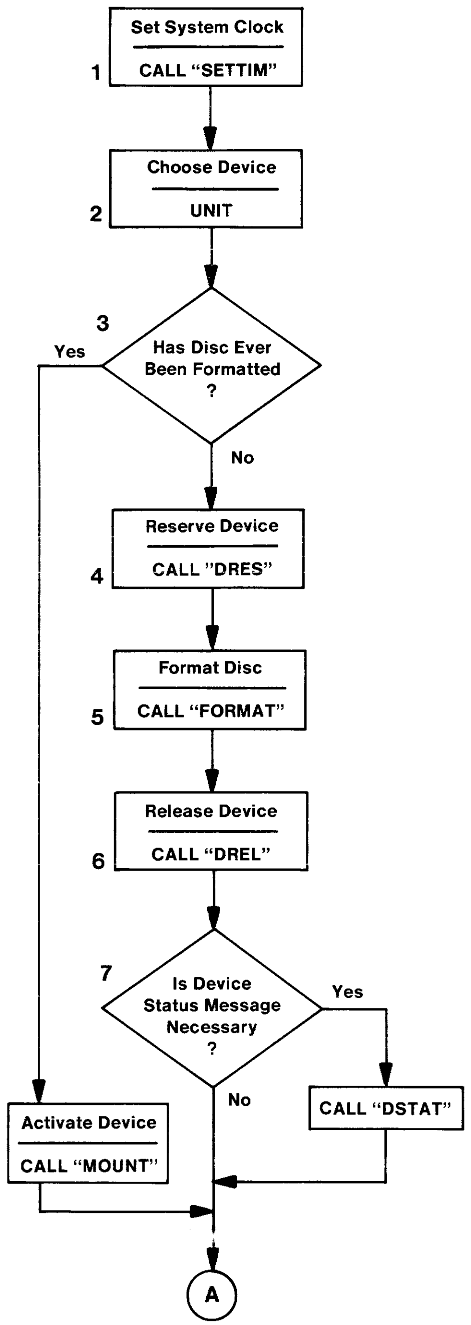
6. Release Device

The reserved device must now be released with a CALL "DREL" command. If the device is not released, no files can be opened.

7. Is Device Status Message Necessary?

If you wish to confirm that the correct disc is in the drive, or what the specifications of that disc are, you may execute a CALL "DSTAT" command. This also allows you to see if the disc has been formatted.

At this point, preparatory work involving the disc is complete. Steps 8 through 16 show commonly used file operations on a disc prepared as shown in steps 1 through 7.



8. Has File Already Been Created?

If you wish to store information and no file has yet been created to receive it, you must execute the CREATE command[a]. This command allows you to specify the name of the file and its attributes, the length of the file, the number of logical records the file will contain, and whether the file is random or sequential.

9. Has File Been Initialized?

Although information may be "randomly" read from a random file, it must be entered sequentially unless the records are first filled with "blank strings." See file initializing program in the section SAMPLE I/O. This program enters blank strings into a file, allowing random data entry.

10. Is File Operation to be SAVE, OLD, APPEND or COPY ... TO?

The above operations may be carried out without the OPEN and CLOSE commands.

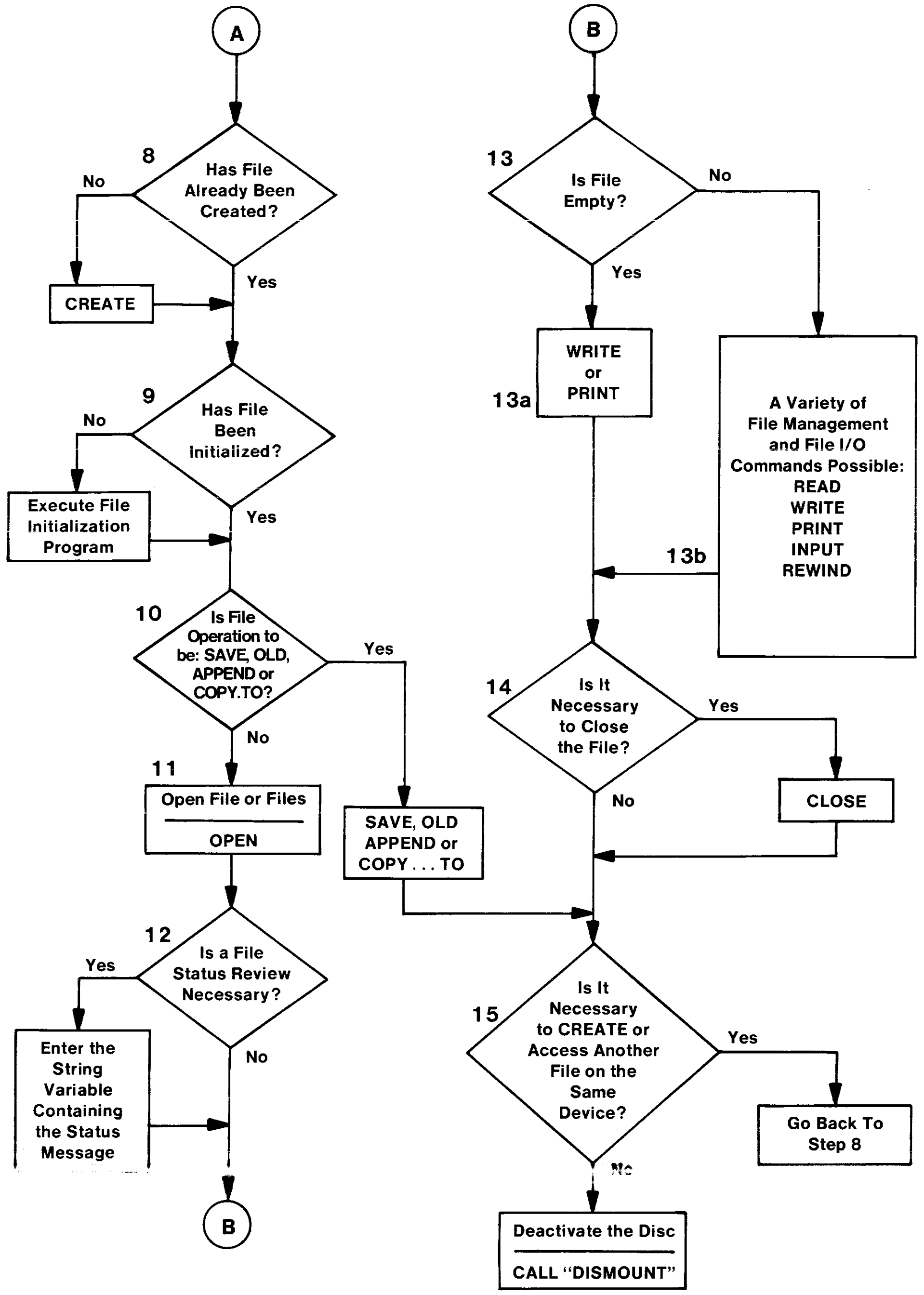
11. Open File or Files

An OPEN command must be executed for those file operations listed in Step 13. This command assigns a logical file number (lfn) to the file, sets file pointers, and generates and stores a file status message for each file opened. Another UNIT command, as in Step 2, is necessary to access a file on a different device.

12. Is a File Status Review Necessary?

It may be necessary to review the status of the file or files just opened. The parameters include the time and date the file was created, altered, and last used, as well as its name, attributes, and space specifications. To see what details are included and how they are displayed, turn to SAMPLE DEVICE AND FILE STATUS MESSAGES. If you do wish to review the file status, enter the string variable that was specified in the last field of the OPEN command.

[a]Except for a SAVE which can create its own file.


A
8
No
Has File Already Been Created?
Yes
CREATE
9
No
Has File Been Initialized?
Yes
Execute File Initialization Program
10
Is File Operation to be: SAVE, OLD, APPEND or COPY.TO?
Yes
No
11
Open File or Files
OPEN
SAVE, OLD APPEND or COPY . . . TO
12
Yes
Is a File Status Review Necessary?
No
Enter the String Variable Containing the Status Message
B
B
13
Is File Empty?
No
Yes
WRITE or PRINT
13a
A Variety of File Management and File I/O Commands Possible:
READ
WRITE
PRINT
INPUT
REWIND
13b
14
Is It Necessary to Close the File?
Yes
No
CLOSE
15
Is It Necessary to CREATE or Access Another File on the Same Device?
Yes
Nc
Go Back To Step 8
Deactivate the Disc
CALL "DISMOUNT"

13. Is File Empty?

If the file has just been created, it contains no information. Information may be entered into the file using a WRITE or PRINT command.

If the file is full or partially full, as indicated in a file status message, you may execute a variety of file management and file I/O operations:

WRITE ... for entering binary information
PRINT ... for entering ASCII information

READ ... for retrieving binary information
INPUT ... for retrieving ASCII information

CALL "REWIND" resets the file pointer to the beginning of the file.

These commands are used to carry out some of the more common file management operations. The command descriptions in Section 5 tell, for each command, if the file must be open or closed for command execution.

14. Is It Necessary to Close the File?

If no further file access is needed, the file can be closed by executing a CLOSE command. This releases the logical file number (lfn) and stops access to the file. It is not necessary to close the file in order to open another, because up to nine files may be opened and used simultaneously. DO NOT CLOSE A SEQUENTIAL FILE OPENED FOR FULL ACCESS WITHOUT WRITING TO IT OR THE FILE CONTENTS WILL BE LOST.

15. Is It Necessary to CREATE or Access Another File on the Same Device?

Your operation may require placing data into or retrieving data from more than one file at a time. To access or create another file on the same device, go back to Step 8. If no further access to this device is desired, it can be deactivated by executing CALL "DISMOUNT." All files must be closed before executing this command. If you want to access a file on a different device, go back to Step 2

Figure 2-6. General Sequence Flow Chart.