

TEKTRONIX®


**4907
FILE MANAGER**

OPERATOR'S MANUAL

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

070-2380-00

First Printing MAR 1978



Copyright © 1978 Tektronix, Inc.

All Rights Reserved.

All software products including this document, all associated tape cartridges and the programs they contain are the sole property of Tektronix, Inc., and may not be used outside the buyer's organization. The software products may not be copied or reproduced in any form without the express written permission of Tektronix, Inc. All copies and reproductions shall be the property of Tektronix and must bear this copyright notice and ownership statement in its entirety.

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

CONTENTS

INTRODUCTION

Section 1	GENERAL DESCRIPTION	Page
	Introduction	1-1
	4907 Configuration	1-1
	4907 Option 30 Configuration	1-2
	4907 Option 31 Configuration	1-2
	Front Panel Controls and Indicators	1-3
	4907 Controller and Single Flexible Disc Drive Cabinet	1-3
	4907 Option 30 Single Flexible Disc Drive Cabinet	1-4
	4907 Option 31 Dual Flexible Disc Drive Cabinet	1-5
	Standard/Optional Accessories	1-6
	4907 Standard Accessories	1-6
	4907 Optional Accessories (All Options)	1-6
	4907 Option 30 Standard Accessories	1-6
	4907 Option 31 Standard Accessories	1-7
	4907F32 Field Update Kit Standard Accessories	1-7
Section 2	GENERAL OPERATION	
	Introduction	2-1
	Preparation	2-1
	GPIB Cable and ROM Pack Installation	2-2
	Power Up	2-3
	4907 (Single Drive)	2-4
	4907 Option 30 and 4907 Option 31	2-4
	Loading the Flexible Disc Drive	2-4
	Flexible Disc Storage and Handling	2-6
	Ready to Start?	2-7
	General Sequence Flow Chart	2-8
	Command Summaries	2-11
	Sample I/O Program	2-15
	Program Description	2-16
	Sample Program Output	2-17
	What Are Random and Sequential Files	2-17
	Sequential Files	2-17
	Random Files	2-18

CONTENTS (cont)

Section 2 (cont)		Page
	Special System Features.....	2-18
	System Clock.....	2-18
	Automatic File Extending.....	2-19
	Status Messages.....	2-19
	Special Characters.....	2-19
	“Group” Open/Next File.....	2-20
	Free Space Message.....	2-20
	Random Access.....	2-20
	Simultaneous File Use.....	2-20
Section 3	STORAGE STRUCTURE	
	Introduction.....	3-1
	What Is a Storage Structure.....	3-1
Section 4	HOW TO WRITE A COMMAND	
	Introduction.....	4-1
	Keyword.....	4-1
	Address.....	4-1
	Argument.....	4-2
	Syntax.....	4-3
	Constant.....	4-3
	String Constant.....	4-3
	Numeric Expression.....	4-3
	Numeric Variable.....	4-4
	String.....	4-4
	String Variable.....	4-4
	Target String Variable.....	4-4
	Target Numeric Variable.....	4-5
	Command Delimiters, Punctuation and Spaces.....	4-5
	Command Examples.....	4-6
	How to Write a File Identifier.....	4-12
	What is a File Identifier?.....	4-12
	File Identifier Construction.....	4-15
	Field Descriptions (Descriptive Form).....	4-16
	Use of 1st Level Libraries.....	4-17
	Examples.....	4-18
	Using Passwords in File Identifiers.....	4-22
	How to Write a Password.....	4-22
	How Much Access?.....	4-24
	Extensions.....	4-25
	File Identifier Delimiters.....	4-26
	General Delimiter Information.....	4-27

CONTENTS (cont)

Section 4 (cont)	Page
Special Characters in File Identifiers	4-27
Special or Multiple File Selection (#)(*)(?)	4-27
Pound Sign (#)	4-28
Asterisk (*)	4-33
Question Mark (?)	4-37
USERLIB String Suppression (@)	4-38
Simplified SYSLIB Access (\$)	4-39

Section 5

COMMAND DESCRIPTIONS

Introduction	5-1
How to Read Command Descriptions	5-1
Format	5-2
Purpose	5-2
Syntax Form	5-2
Descriptive Form	5-3
Field Definitions	5-3
General Information	5-3
Prerequisites	5-3
Examples	5-3
System Commands	5-5
DELALL (Delete All)	5-5
CALL "FMVALS" (File Manager Values)	5-6
INIT (Initialize)	5-8
CALL "SETTIM" (Set Time)	5-9
CALL "TIME"	5-11
UNIT	5-12
CALL "UNIT"	5-14
CALL "USERLIB"	5-16
Controller/Device Commands	5-19
CALL "COMPRS" (Compress)	5-19
CALL "CUSTAT" (Controller Unit Status)	5-21
CALL "DISMOUNT"	5-23
CALL "DREL" (Device Release)	5-25
CALL "DRES" (Device Reserve)	5-27
CALL "DSTAT" (Device Status)	5-29
CALL "FFRMT" (Fast Format)	5-31
CALL "FORMAT"	5-32
CALL "HERRS" (Hard Error Status)	5-36
CALL "MOUNT"	5-38
CALL "MRKBBG" (Mark Bad Block Group)	5-40

CONTENTS (cont)

Section 5 (cont)		Page
	File Management Commands.....	5-43
	ASSIGN	5-43
	CLOSE.....	5-45
	COPY . . . TO.....	5-47
	CREATE.....	5-54
	DIRECTORY.....	5-59
	CALL "DUP" (Duplicate).....	5-62
	END.....	5-64
	CALL "FILE".....	5-65
	KILL.....	5-67
	CALL "NEXT".....	5-69
	ON EOF (On End-of-File).....	5-71
	OPEN	5-73
	CALL "RENAME"	5-78
	CALL "REWIND".....	5-81
	SECRET.....	5-83
	CALL "SPACE"	5-84
	TYP (TYPE).....	5-86
	File I/O Commands	5-89
	APPEND.....	5-89
	INPUT.....	5-92
	OLD.....	5-95
	PRINT.....	5-97
	READ	5-101
	SAVE.....	5-104
	WRITE	5-106
Section 6	SPECIFICATIONS	
	4907 Performance Specifications	6-1
	4907 Physical Specifications	6-1
	4907 Environmental Specifications.....	6-2
	4907 Electrical Specificatons.....	6-3
	Flexible Disc Drive Specifications	6-3
	Media Requirements.....	6-4
	ROM Pack	6-5
Appendix A	DEVICE AND FILE STATUS MESSAGES	
Appendix B	ERROR MESSAGES AND RECOVERY PROCEDURES	
Appendix C	FILE POINTER OPERATION	

CONTENTS (cont)

Appendix D	GLOSSARY
Appendix E	ROUTINE FLEXIBLE DISC DRIVE MAINTENANCE
Appendix F	SAMPLE PROGRAMS
Appendix G	USING PRINT AND INPUT IN A PROGRAM
Appendix H	USING WRITE AND READ IN A PROGRAM
INDEX	

TABLES

Table	Description	Page
5-1	Data Type (Message From TYPE Function)	5-87
6-1	Line Voltages	6-3
C-1	Sequential Files Pointer Location	C-4
C-2	Random Files Pointer Location	C-4

ILLUSTRATIONS

Figure	Description	Page
1-1	4907 (Single Drive)	xiv
1-2	4907 Option 30 (Two Drives)	xiv
1-3	4907 Option 31 (Three Drives)	xiv
1-4	Diagram of 4907 Configuration	1-1
1-5	Diagram of 4907 Option 30 Configuration	1-2
1-6	Diagram of 4907 Option 31 Configuration	1-2
1-7	Front Panel Controls and Indicators for 4907 Main Cabinet	1-3
1-8	Front Panel Controls and Indicators for 4907 Option 30 Single Drive Cabinet	1-4
1-9	Front Panel Controls and Indicators for 4907 Option 31 Dual Drive Cabinet	1-5
2-1	Connecting GPIB Cable to Rear of Main 4907 Cabinet	2-2
2-2	Plugging in ROM Pack	2-3
2-3	Placing Flexible Disc in Drive	2-4
2-4	Closing Drive Door	2-5
2-5	General Sequence Flow Chart	2-8
2-6	General Sequence Flow Chart (cont)	2-9
2-7	General Sequence Flow Chart (cont)	2-10
3-1	Sample Storage Structure (1 Level)	3-1
3-2	Sample Storage Structure (1 Level)	3-2
3-3	Sample Storage Structure (2 Levels)	3-2
3-4	Sample Storage Structure (5 Levels)	3-3
4-1	Sample Storage Structure (1 Level, 1 File)	4-12
4-2	Sample Storage Structure (5 Levels, 1 File)	4-13
4-3	Sample Storage Structure (5 Levels, 4 Files)	4-14
4-4	Syntax and Descriptive Form of F.I. (File Identifier) Construction	4-15
4-5	Sample Storage Structure (5 Levels, 7 Files)	4-19
4-6	Sample Storage Structure (4 Levels, 7 Files)	4-20
4-7	Sample Storage Structure (4 Levels, 5 Files)	4-21
4-8	Sample Storage Structure (2 Levels, 1 File)	4-23
4-9	Sample Storage Structure (2 Levels, 2 Files, 1 With Password)	4-23
4-10	Sample Storage Structure (5 Levels, 6 Files)	4-25
4-11	Sample Storage Structure (2 Levels)	4-28
4-12	Sample Storage Structure (3 Levels)	4-29
4-13	Sample Storage Structure (4 Levels)	4-30
4-14	Sample Storage Structure (5 Levels)	4-31
4-15	Sample Storage Structure (5 Levels)	4-32

ILLUSTRATIONS (cont)

Figure	Description	Page
4-16	Sample Storage Structure (4 Levels, 8 Files)	4-34
4-17	Sample Storage Structure (3 Levels, 7 Files)	4-35
5-1	Sample Storage Structure (4 Levels, 3 Files)	5-76
5-2	Sample Storage Structure (4 Levels, 5 Files)	5-77
5-3	Sample Storage Structure Illustrating How to Change File Names	5-79
5-4	Sample Storage Structure Illustrating How to Transfer Files From One Library to Another	5-80
C-1	Graphic Representation of Files	C-1
C-2	Pointer location at the end of data entry after an OPEN "F"	C-1
C-3	Pointer location at the end of data entry after an OPEN "U"	C-2
C-4	Pointer location prior to I/O command after an OPEN "R"	C-2
C-5	Pointer Location after Data Entry Before CALL "REWIND"	C-3
C-6	Pointer Location after CALL "REWIND"	C-3
E-1	Removing Main 4907 Cabinet Cover	E-2
E-2	Cleaning Read/Write Head	E-3
E-3	Removing Main 4907 Cabinet Cover	E-4
E-4	Removing Read/Write Head Button	E-4
E-5	Installing New Read/Write Head Button	E-5

INTRODUCTION

Until now, 4051 Graphic System users have relied on magnetic tape for data and program storage. Now, however, the 4907 File Manager (incorporating the 4051, the 4907 ROM pack and the 4907 disc drives) changes that. The changes include much more versatile file management, faster file access, and increased storage capacity. Much of the improvement is due to the 4907's flexible magnetic discs, but there is more to it than that.

Improved file management is due also to the many new "ROM pack" commands added to the 4051's BASIC vocabulary. These commands, when used with regular 4051 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 definitely increases the potential and versatility built into your 4051. We believe you'll find this addition to the 4051 product line a valuable and efficient tool in all your information storage activities.

ABOUT THIS MANUAL

This manual explains how to operate the 4907 File Manager after the procedures outlined in the 4907 Installation Guide have been carried out.

In order to use this manual and the 4907 File Manager effectively, you must have a working knowledge of the 4051 Graphic System. Most of the descriptions, procedures and terms in this manual have been written on that basis.

Those aspects of data transfer (I/O) and related subjects not covered in this manual are fully described in the 4051 Graphic System Reference Manual.

INTRODUCTION

Your 4051 Graphic System and 4907 disc drives, together, are referred to as "the system" throughout this manual. Your 4051 is often called a "host." Although the 4051 is generally purchased separately, it is considered part of "the system" for purposes of discussion.

HOW TO USE THE MANUAL

This manual is written to help you create and use disc files effectively and efficiently.

We suggest that you take time to familiarize yourself with the following sections of the manual before attempting to operate the system.

1. FRONT PANEL CONTROLS AND INDICATORS

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

2. GENERAL OPERATION

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. This section contains command summaries that show the general construction and provide a brief description of each command. Review the GLOSSARY (Appendix D) to understand the terms used.

3. STORAGE STRUCTURE

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.

4. HOW TO WRITE A COMMAND

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

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 4051 which file is involved in the operation and where it is to be created or where it can be found.

5. COMMAND DESCRIPTIONS

4907 File Manager commands are categorized and 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 **WHAT ARE RANDOM AND SEQUENTIAL FILES** in **GENERAL OPERATION** (Section 2). Also see the descriptions of **WRITE** and **PRINT** in the 4051 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 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.

INTRODUCTION

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 illegal or syntactically 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 4051
- not executing an INIT when necessary

In case of problems, check the sequences required as well as the individual operations. Be sure the problem is not caused by incorrect use of the 4051 or faulty programming.

NOTE

If an undefined variable is entered in any CALL command, an error message appears that advises you of an illegal command (instead of an undefined variable message). No error message occurs if the field is defined as a target string variable in the command description.

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

A. See FILE I/O COMMANDS in Section 5. Also see SAMPLE I/O PROGRAM in Section 2.

Q. WHAT ABOUT PROGRAMMING?

- A. All normal 4051 programming rules apply. No regular 4051 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 PROGRAM in the appendices.

Q. WHAT ABOUT EXTERNAL DEVICES SUCH AS INSTRUMENTATION AND 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 host (4051).

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 B).

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

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

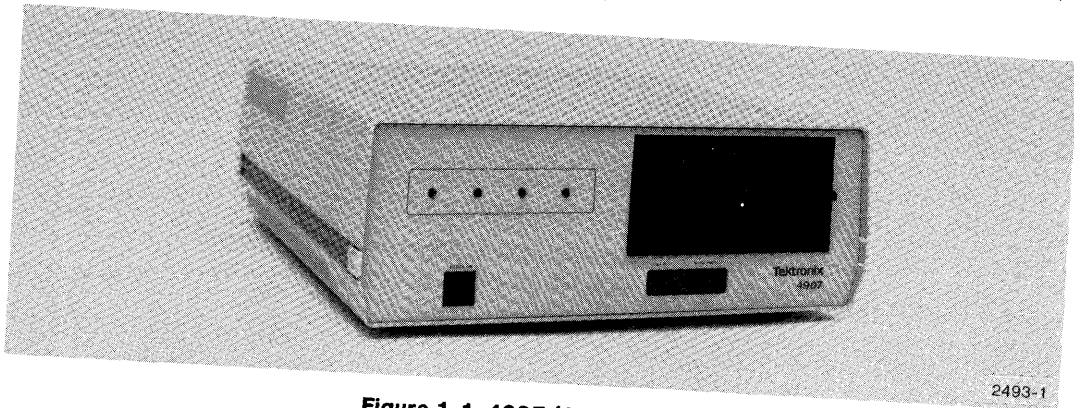


Figure 1-1. 4907 (Single Drive).

2493-1

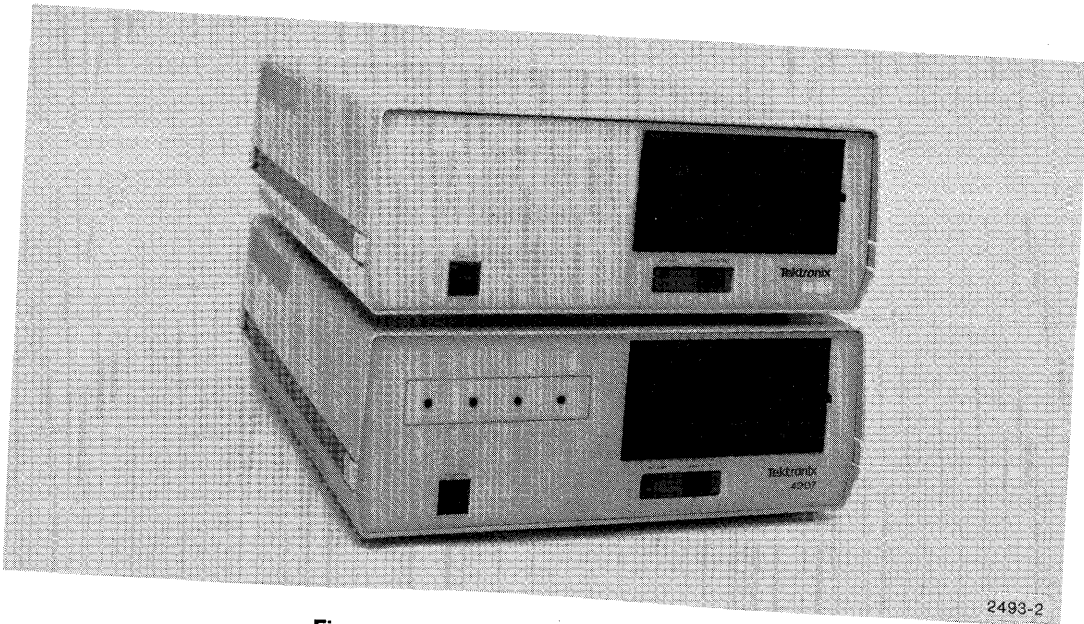


Figure 1-2. 4907 Option 30 (Two Drives).

2493-2



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

2493-3

SECTION 1

CONTENTS

	Page
Introduction	1-1
4907 Configuration	1-1
4907 Option 30 Configuration	1-2
4907 Option 31 Configuration	1-2
Front Panel Controls and Indicators	1-3
4907 Controller and Single Flexible Disc Drive Cabinet	1-3
4907 Option 30 Single Flexible Disc Drive Cabinet	1-4
4907 Option 31 Dual Flexible Disc Drive Cabinet	1-5
Standard/Optional Accessories	1-6
4907 Standard Accessories	1-6
4907 Optional Accessories (All Options)	1-6
4907 Option 30 Standard Accessories	1-6
4907 Option 31 Standard Accessories	1-7
4907F32 Field Update Kit Standard Accessories	1-7

Section 1

GENERAL DESCRIPTION

INTRODUCTION

All 4907 File Managers are designed for use with a 4051 Graphic System, which controls all storage activity.¹

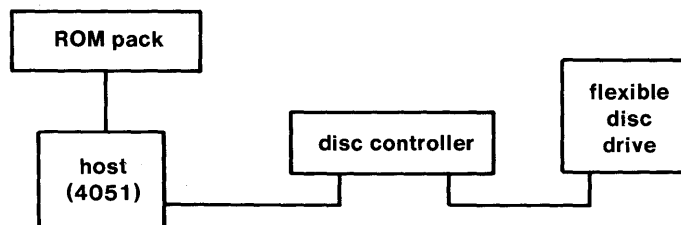
The 4907 File Manager includes a 4907 ROM (Read-Only Memory) pack and a 4907. The 4907 contains a disc controller and from one to three flexible disc drives.

The ROM pack, which plugs into the 4051, provides extra BASIC commands for use by the 4051. These commands are transmitted from the 4051 keyboard or program through a GPIB cable to the 4907 disc controller in an adjacent 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 is shipped in one of three configurations:

1. 4907 (single disc drive).
2. 4907 Option 30 (two disc drives).
3. 4907 Option 31 (three disc drives).

4907 CONFIGURATION

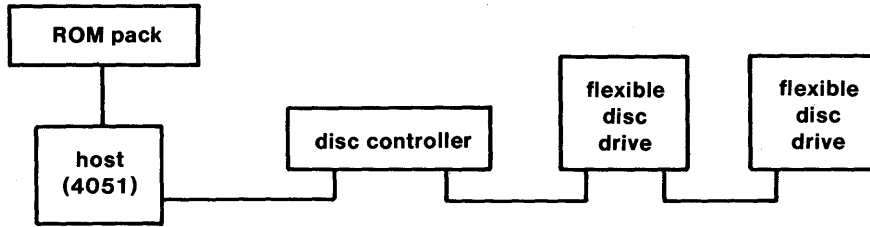


2380-1

Figure 1-4. Diagram of 4907 Configuration.

¹Except for front panel switches on disc drives and disc controller.

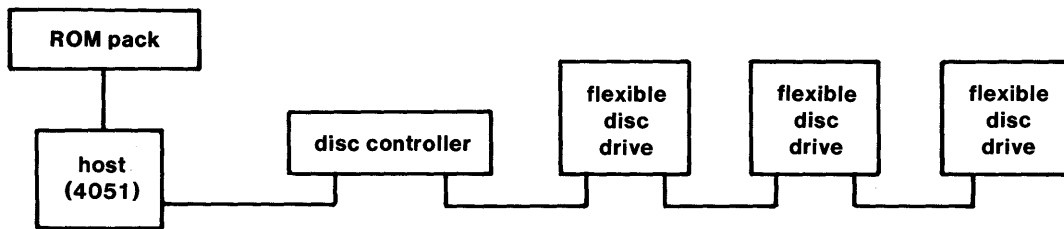
4907 OPTION 30 CONFIGURATION



2380-2

Figure 1-5. Diagram of 4907 Option 30 Configuration.

4907 OPTION 31 CONFIGURATION

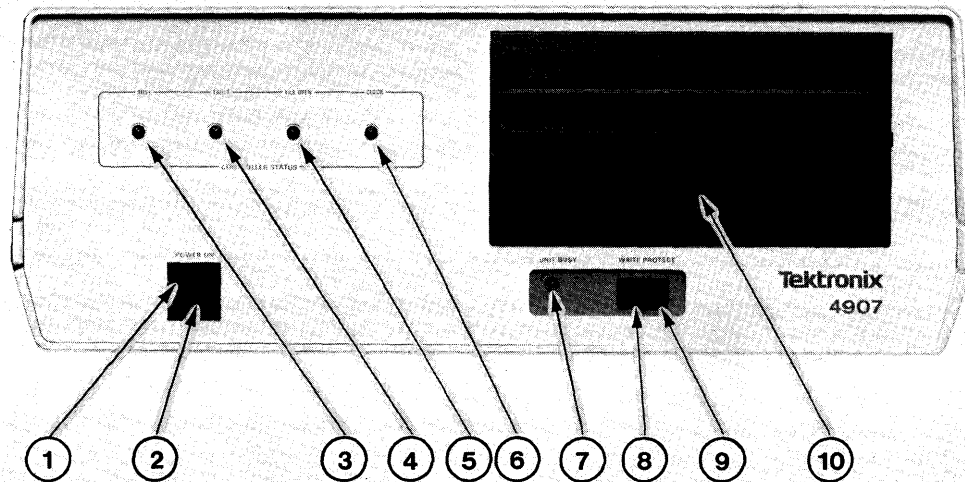


2380-3

Figure 1-6. Diagram of 4907 Option 31 Configuration.

FRONT PANEL CONTROLS AND INDICATORS

4907 Controller and Single Flexible Disc Drive Cabinet



2380-4

Figure 1-7. Front Panel Controls and Indicators for 4907 Main Cabinet.

1. Power indicator.
2. Power switch.
3. Busy indicator. If lit, disc operation is being carried out.
4. Fault indicator. If lit, system is inoperative. If restart efforts fail, see 4907 Service Manual or contact a Tektronix technician.
5. File open indicator. If lit, 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 lit, system clock must be set.
7. Busy indicator. If lit, disc operation is being carried out.
8. Write protect switch.²
9. Write protect indicator. If lit, device or flexible disc is in the write-protect state.
10. Drive door release.

²Disc may be write-protected independent of switch setting.

GENERAL DESCRIPTION

4907 Option 30 Single Flexible Disc Drive Cabinet

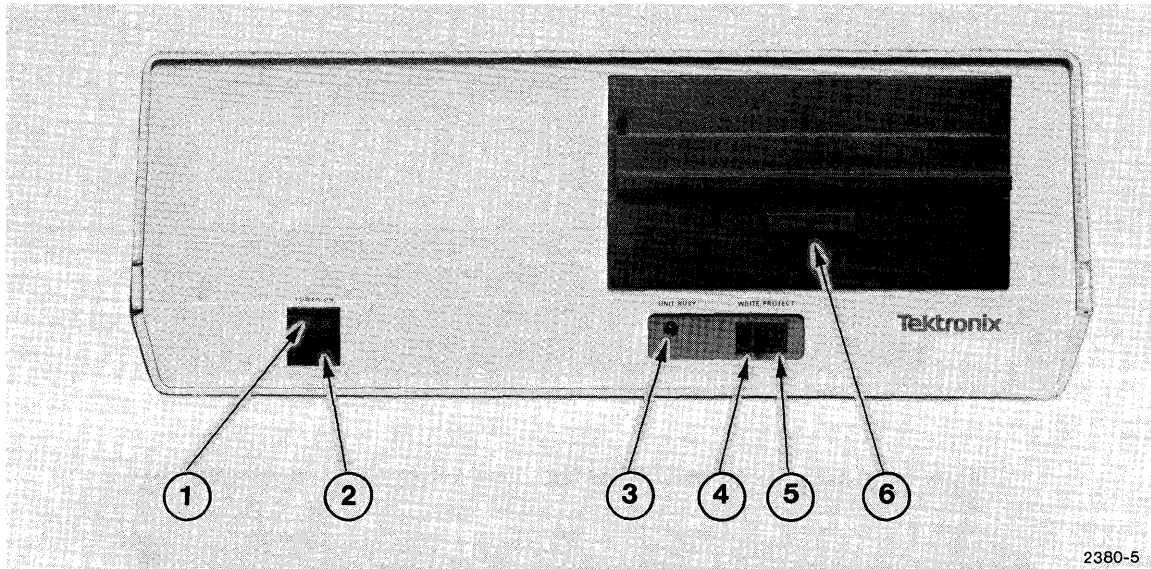


Figure 1-8. Front Panel Controls and Indicators for 4907 Option 30 Single Drive Cabinet.

1. Power indicator.³
2. Power switch.
3. Busy indicator. If lit, disc drive operation is being carried out.
4. Write-protect switch.⁴
5. Write-protect indicator. If lit, device or flexible disc is in the write-protect state.
6. Drive door release.

³Even though indicator may be lit, the unit becomes inoperative if power switch on the main cabinet is turned off.

⁴Disc may be write-protected independently of switch setting.

4907 Option 31 Dual Flexible Disc Drive Cabinet

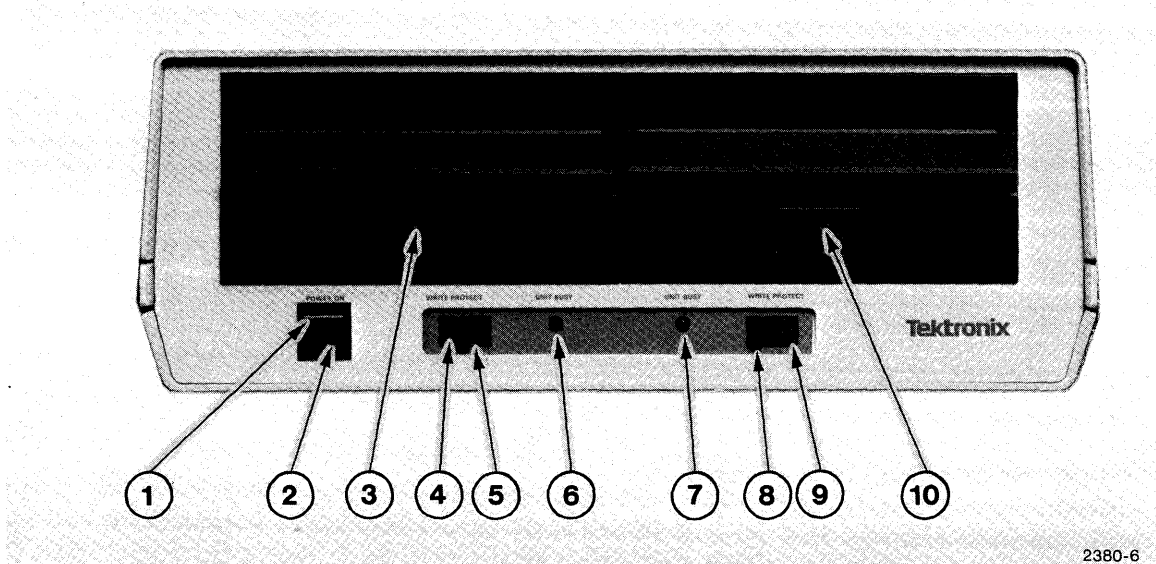


Figure 1-9. Front Panel Controls and Indicators for 4907 Option 31 Dual Drive Cabinet.

1. Power indicator.³
2. Power switch for both drives.
3. Left drive door release.
4. Write-protect indicator for left drive.⁵
5. Write-protect switch for left drive.⁶
6. Busy indicator. If lit, left drive operation is being carried out.
7. Busy indicator. If lit, right drive operation is being carried out.
8. Write-protect switch for right drive.⁶
9. Write-protect indicator for right drive.⁵
10. Right drive door release.

⁵If lit, device or flexible disc is in the write-protect state.

⁶Disc may be write-protected independent of switch setting.

GENERAL DESCRIPTION

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 Media
- Box of Cleaning Pads (10)
- GPIB Cable 2M
- 4907 Installation Guide
- 4907 File Manager Pocket Reference Card
- 4907 File Manager ROM Pack
- 4907 File Manager Label Set

4907 Optional Accessories (All Options)

- Box of Flexible Disc Media (10)
- 4907 Service Manual
- GPIB Cable 4M
- Flexible Disc Drive Service Manual
- Alignment Disc

4907 Option 30 Standard Accessories

- Power Cord
- Interconnect Cable
- Strain Relief Bracket
- Clamp
- 1 Flexible Disc Media
- Box of Cleaning Pads (10)
- 4907 Installation Guide
- 4907 File Manager Label Set

4907 Option 31 Standard Accessories

- Power Cord
- Interconnect Cable
- Strain Relief Bracket
- Clamp
- 2 Flexible Disc Media
- Box of Cleaning Pads (10)
- 4907 Installation Guide
- 2 4907 File Manager Label Sets

4907F32 Field Update 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.

- 1 Flexible Disc
- Box of Cleaning Pads (10)
- 4907 Installation Guide
- 4907 File Manager Label Set

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

	Page
Introduction	2-1
Preparation	2-1
GPIB Cable and ROM Pack Installation	2-2
Power Up	2-3
4907 (Single Drive).....	2-4
4907 Option 30 and 4907 Option 31	2-4
Loading the Flexible Disc Drive	2-4
Flexible Disc Storage and Handling	2-6
Ready to Start?	2-7
General Sequence Flow Chart	2-8
Command Summaries	2-11
Sample I/O Program	2-15
Program Description.....	2-16
Sample Program Output	2-17
What Are Random and Sequential Files	2-17
Sequential Files.....	2-17
Random Files	2-18
Special System Features.....	2-18
System Clock	2-18
Automatic File Extending	2-19
Status Messages.....	2-19
Special Characters.....	2-19
"Group" Open/Next File	2-20
Free Space Message	2-20
Random Access.....	2-20
Simultaneous File Use.....	2-20

Section 2

GENERAL OPERATION

INTRODUCTION

This section outlines the steps required before the system 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. This section contains command summaries that illustrate the general construction and provide a brief description of each command. 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, any necessary address strap and line voltage changes must be made. The line cord or cords, ribbon cable,¹ ROM pack, and GPIB cable must then be installed. The system is then powered up, the discs loaded, and performance checks carried out. All these procedures are described in the 4907 Installation Guide.

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.

¹ Not applicable to the single disc drive 4907.

GENERAL OPERATION

GPIB Cable and ROM Pack Installation

1. Turn off 4051.
2. Connect the GPIB cable from the rear of the 4051 to the rear of the main 4907 cabinet (Figure 2-1).

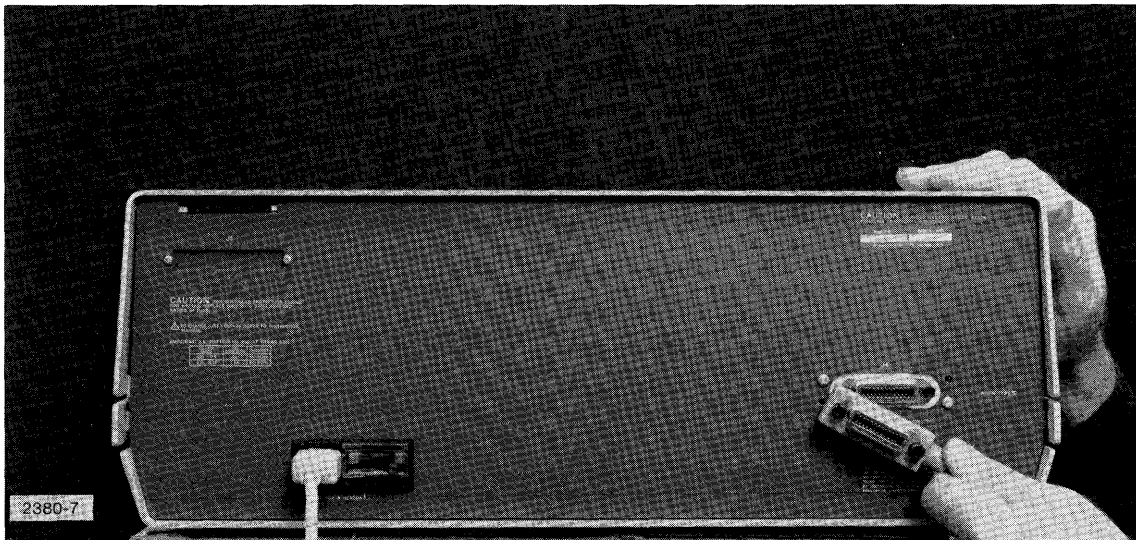


Figure 2-1. Connecting GPIB Cable to Rear of Main 4907 Cabinet.

3. Plug the 4907 File Manager ROM pack into any available slot in the 4051 back pack (Figure 2-2) or into a ROM Expander. BE SURE THE 4051 HAS BEEN TURNED OFF.

CAUTION

Inserting any device into or removing any device from a 4051 backpack slot or a ROM Expander slot when power is on may cause memory to be erased.

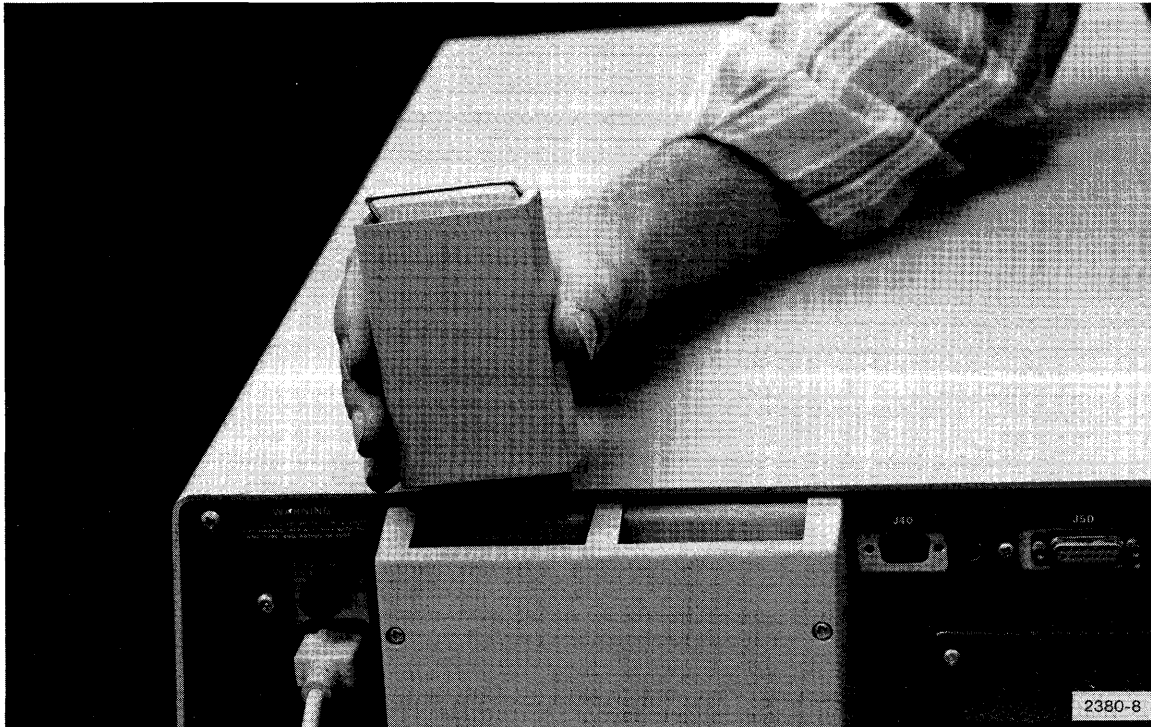


Figure 2-2. Plugging in ROM Pack.

CAUTION

The ROM pack may also be plugged into any available slot in any 405IEO1 ROM Expander. Only one 4907 File Manager ROM pack may be plugged into either the ROM expander or the backpack on the same 4051. Installing more than one of these ROM packs sets up a conflict since each 4907 would be considered GPIB device No. 0.

4. The 4907 can now be powered up. See POWER UP.

Power Up

CAUTION

If multiple GPIB devices are interfaced to the 4051 at least half of them plus one must be turned on before turning on the 4051. If this is not done the I/O indicator may light continuously, indicating a system hang-up. The devices must be turned on even if they are not going to be used. The alternative is to disconnect them entirely.

GENERAL OPERATION

4907 (Single Drive)

1. Turn on front panel power switch on main 4907 cabinet.
2. Turn on 4051.

4907 Option 30 and 4907 Option 31

1. Turn on front panel power switch on main 4907 cabinet.
2. Turn on front panel power switch on auxiliary cabinet.
3. Turn on 4051.

Loading the Flexible Disc Drive

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 far in as it will go. Be sure the tape supplied with the disc is covering the write-protect hole whenever formatting or writing to a disc. To write-protect a disc, remove the tape covering the write-protect hole.

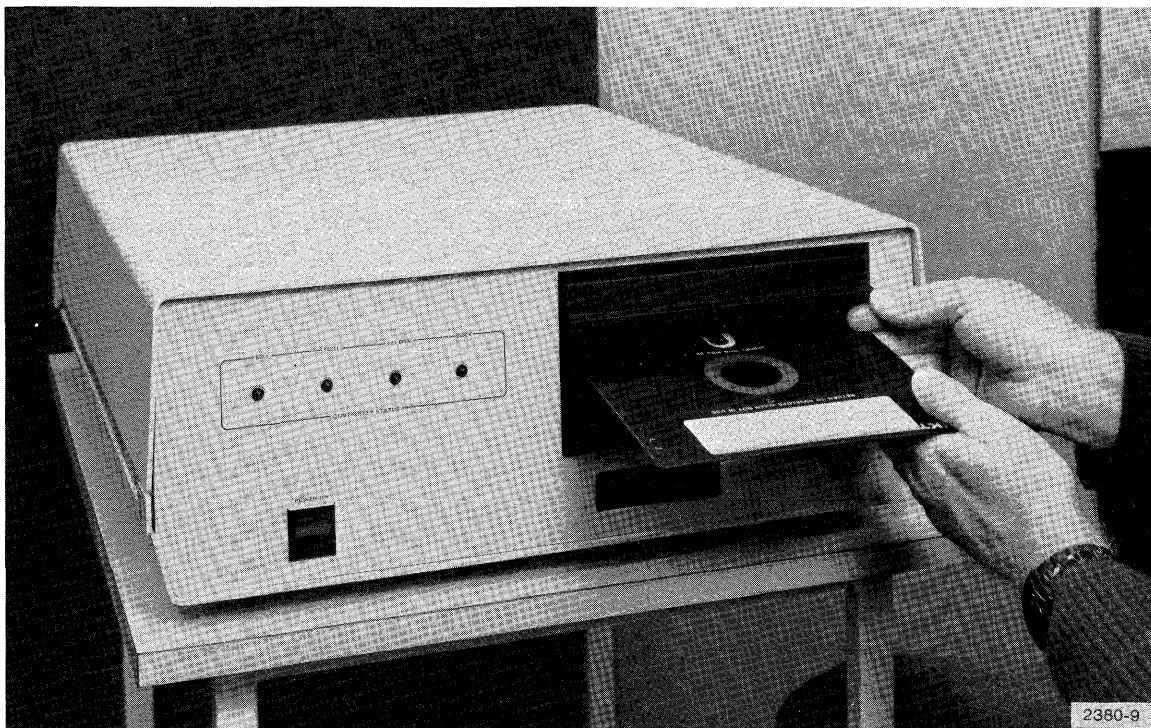


Figure 2-3. Placing Flexible Disc in Drive.

2. Close door (Figure 2-4).

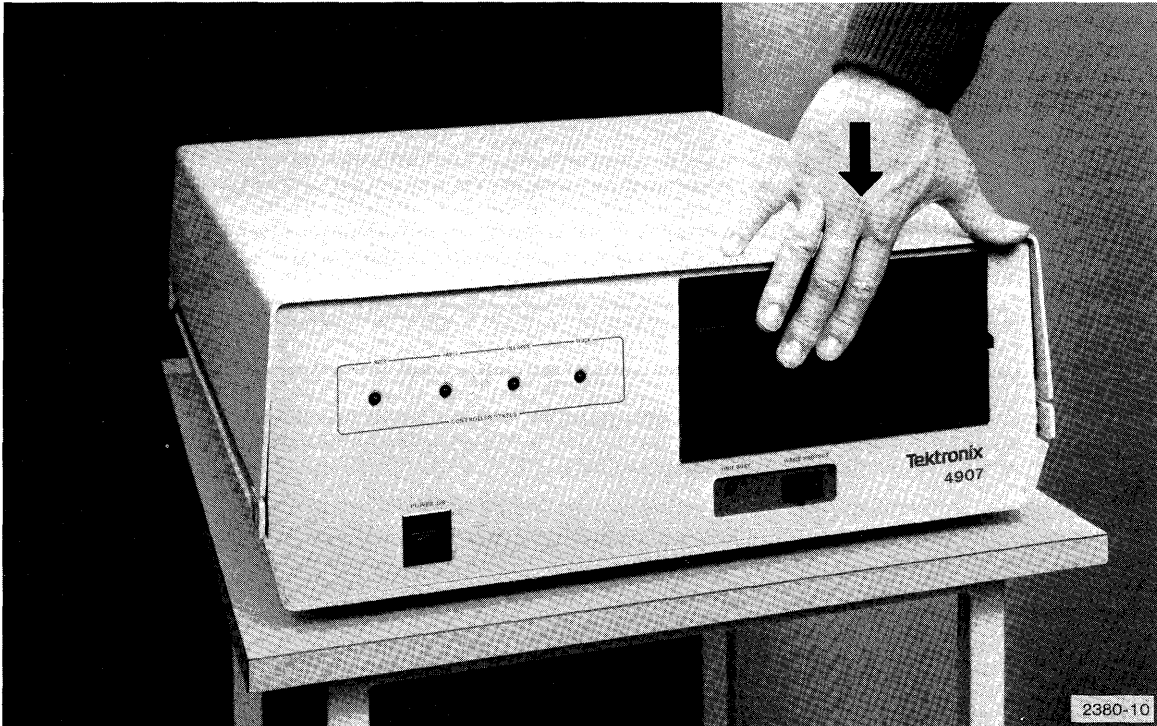


Figure 2-4. Closing Drive Door.

3. Repeat sequence with additional drives, if necessary.
4. Start operation.

Flexible Disc Storage and Handling

- Do not place heavy objects on disc cartridge.
- Keep fingers off disc surface.
- Do not write directly on cartridge but use labels instead.
- Attempts to clean the disc may damage the surface.
- Keep disc away from heat and sunlight.
- Allow a disc at least five minutes to adjust to existing temperature and humidity before using.
- Store discs vertically to prevent dust from settling on disc surfaces.
- Keep cartridges away from magnetic fields and magnetic material. Magnetism can destroy stored data.

READY TO START?

At this point you probably know what kind of information you want to store but not the procedures necessary to store it.

The GENERAL SEQUENCE FLOW CHART shows the steps necessary in creating and using files. By reading the accompanying text you'll get a good idea of the general requirements.

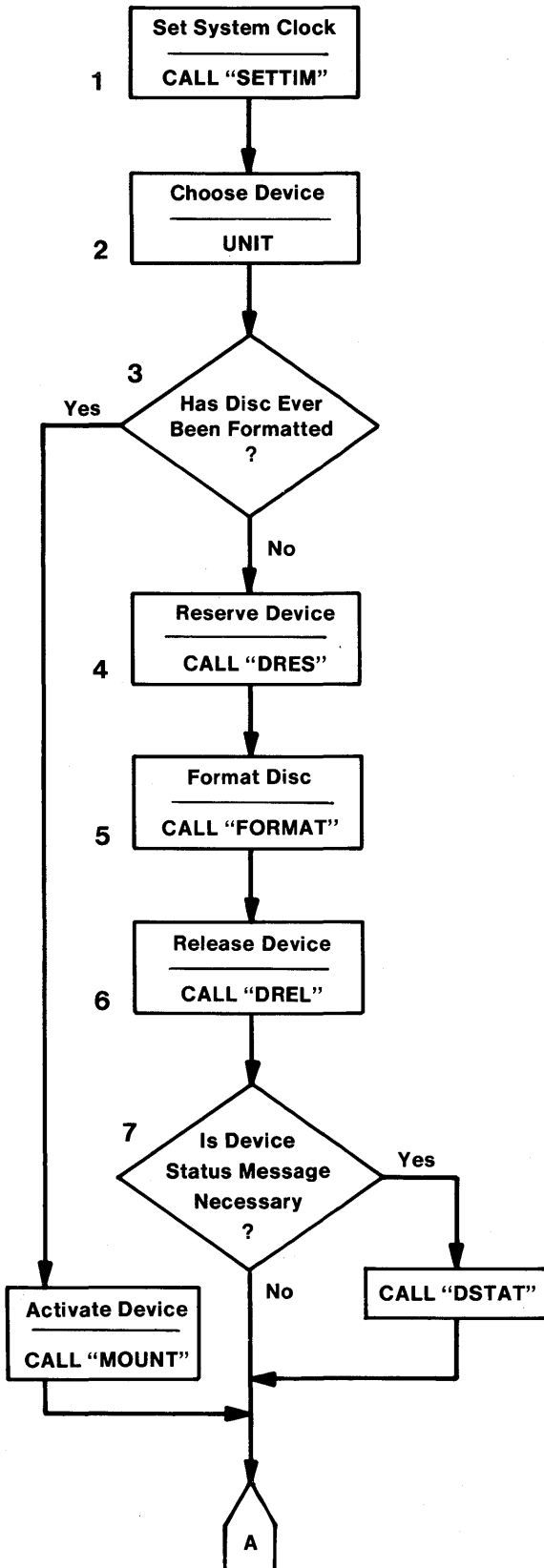
Occasionally, commands not shown in the flow chart will be necessary; for example, you may wish to change the name or attributes of a file or reduce allocated file space. For a quick look at all system commands, what they will do and their general construction, see COMMAND SUMMARIES in this section. For more detailed information see COMMAND DESCRIPTIONS (Section 5).

A small program demonstrating how to create a random and sequential file is included in this section (SAMPLE I/O PROGRAM). This program also shows how to write binary information and print ASCII information to these files as well as how to recover it. The differences between random and sequential files are described in WHAT ARE RANDOM AND SEQUENTIAL FILES?

The 4907 File Manager offers features normally not found in other storage systems. These features and the associated commands are discussed in SPECIAL SYSTEM FEATURES in this section.

Naturally there is more to operating the 4907 File Manager than jockeying commands around. How you write those commands will make the difference in your use of system capabilities. To write effective commands you should become familiar with storage concepts and "file identifiers." See STORAGE STRUCTURE (Section 3) and HOW TO WRITE A FILE IDENTIFIER in Section 4.

GENERAL SEQUENCE FLOW CHART



1. Set System Clock

The CALL "SETTIM" command starts the system clock so the time and date of various system activities are recorded. This command is required each time the system is turned off and restarted. There is no "default" time provided. This command is mandatory.

2. Choose Device

The UNIT command identifies which device is to be the "current" device, the one the system accesses 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

The device must be reserved with CALL "DRES" before any preparatory disc operation, such as 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.

GENERAL SEQUENCE FLOW CHART (cont)

At this point, preparatory work involving the disc is complete. Sequences 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, the CREATE command must be executed.¹ 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 PROGRAM. This program enters blank strings into a file, allowing random data entry.

10. Is File Operation to be SAVE, OLD, APPEND or COPY . . . TO?

These operations may be carried out without the OPEN and CLOSE commands.

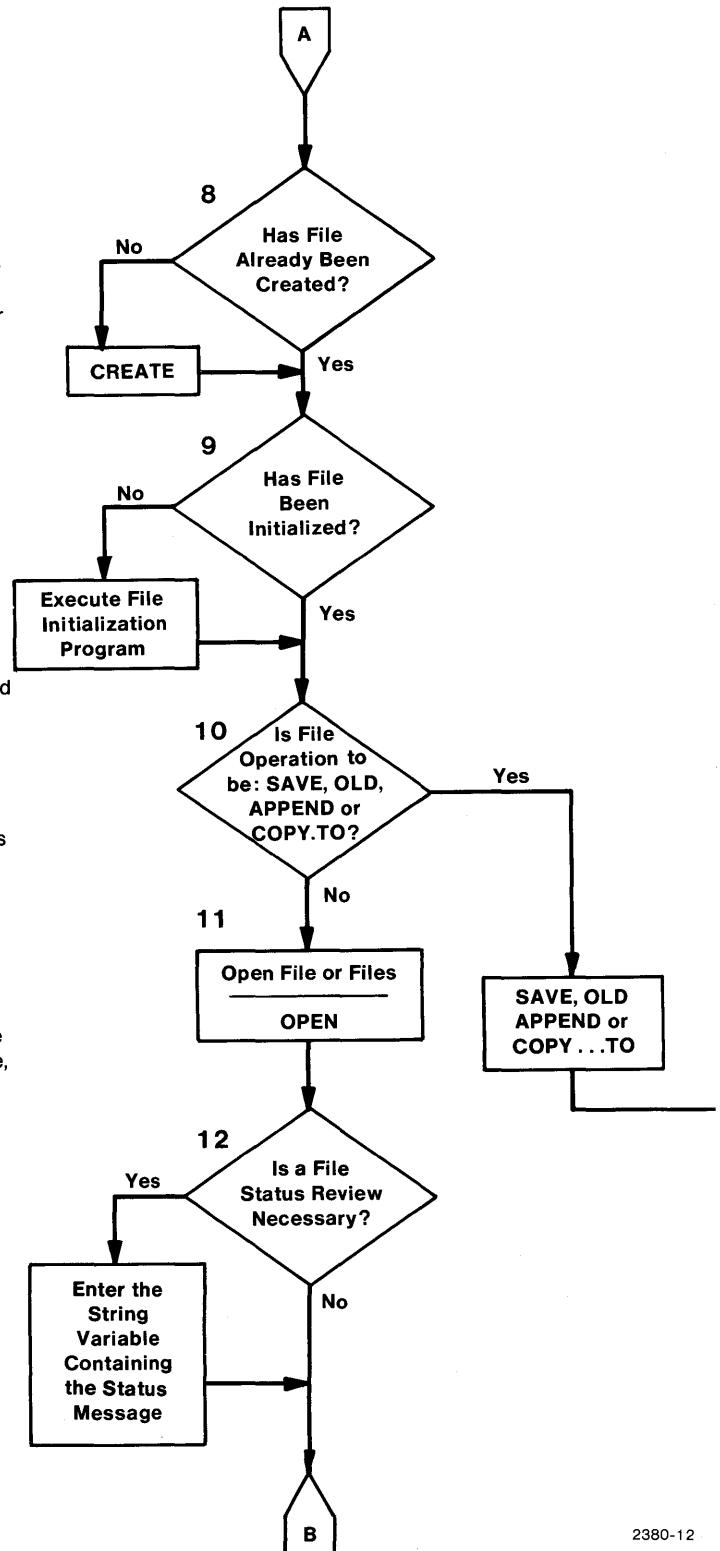
11. Open File or Files

An OPEN command must be executed for those file operations listed in Step 13. This command also 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.

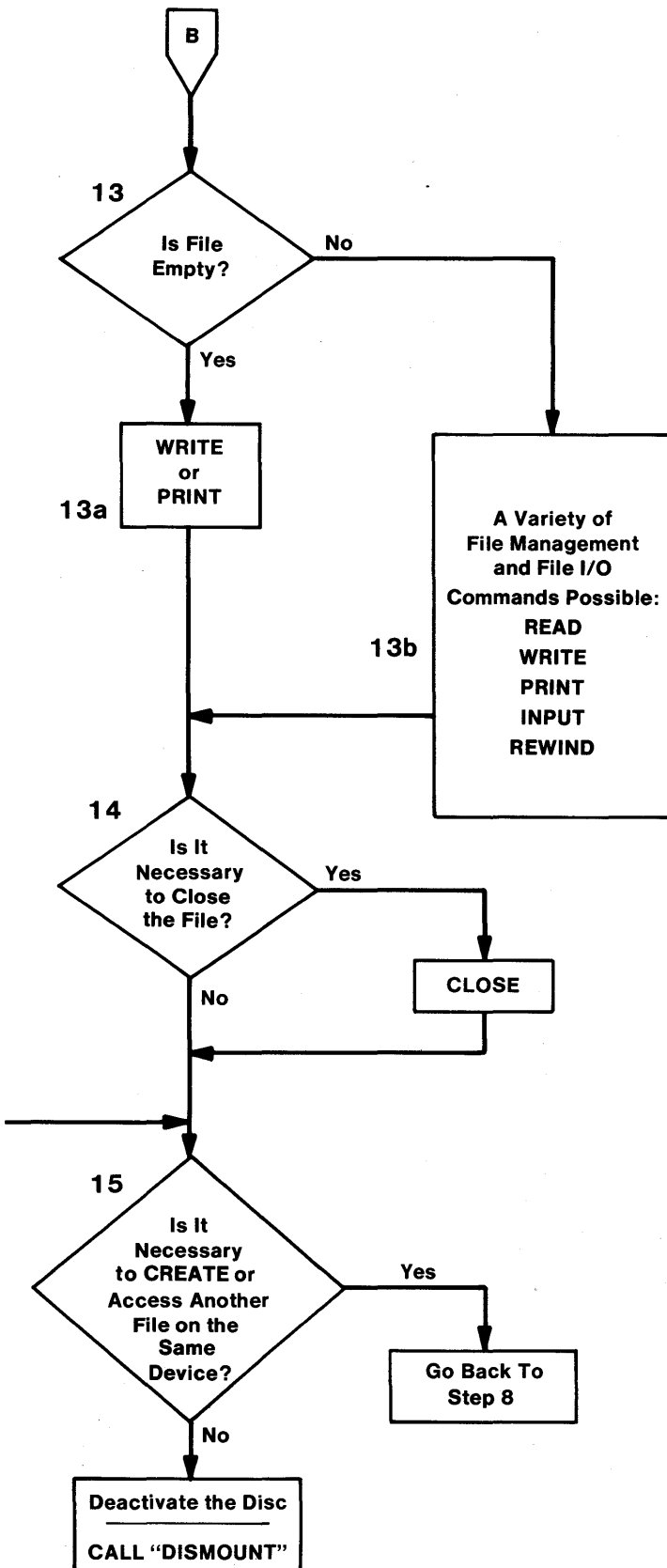
¹Except for a SAVE which can create its own file.



2380-12

Figure 2-6. General Sequence Flow Chart.

GENERAL SEQUENCE FLOW CHART (cont)



13. Is File Empty?

If the file has just been created, it contains no information. Information now 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, a variety of file management and file I/O operations may be executed. For example, the following commands are now possible:

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 examples in How to Write a Command, in the 4907 File Manager Operator's Manual, outline other operations that are less common.

14. Is It Necessary to Close the File?

If no further file access is necessary, the file may be closed by executing a CLOSE command. This automatically releases 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.

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. If you wish to access or create another file on the same device, go back to Step 8. If no further access to this device is necessary, it may be dismounted by executing CALL "DISMOUNT". All files must be closed before executing this command. If you wish to access a file on a different device, go back to Step 2.

Command Summaries

Information in parentheses helps describe the preceding field (except in TYPE and ON EOF summaries where the parentheses are actually part of the syntax). Ifn means "logical file number." Remember that all F.l.s (File Identifiers) must be within quotation marks.

NAME (KEYWORD) AND GENERAL FORMAT	FUNCTION
APPEND F.I.["ASCII"]; target line number[,increment between line numbers]	Adds the program in specified file to memory at the stated line number.
ASSIGN F.I.;attributes (R,U,S,C,N,M)	Changes file attributes such as: R Private U Public S Scattered C Contiguous N Not Compressible M Compressible
CLOSE [Ifn]	Closes one or all files.
CALL "COMPRS", device address, compress control (1 or 0)	Collects all scattered free space on the disc into one contiguous space.
COPY . . . F.I. #1 ,source device address TO F.I. #2 ,target device address	Copies files to the same or another disc.
CREATE F.I. [,attributes]; number of logical records, record length	Defines a new empty file.
CALL "CUSTAT", target string variable	Generates a device status message for each device interfaced to the 4907 File Manager controller and stores them in the specified string variable.
DELALL	Erases everything currently in 4051 memory and closes all files.

GENERAL OPERATION

DIRECTORY [I/O address:] [format code (0, 1, or 2) [F.I.]]	<p>Lists (to 4051 screen or mag tape) facts about specified files in storage.</p> <p>Depending on the format code entry, listings can include:</p> <p>0 File names only. 1 File names and dates. 2 Full file status.</p>
CALL "DISMOUNT", device address	Deactivates specified device when all open files are closed. Device remains active until all files are closed. Until files are closed, I/O may continue but no new file can be opened.
CALL "DREL", device address	Releases a reserved disc drive.
CALL "DRES", device address	Reserves a disc drive.
CALL "DSTAT", device address, target string variable	Generates a status message about the disc in the drive or the device specified.
CALL "DUP", source device address, target device address, compress control (0 or 1)	Duplicates contents from one disc to another.
END	Stops program execution and closes files.
CALL "FFRMT",	Initializes a disc without searching first for defective sectors. See CALL "FORMAT".
CALL "FILE", device address, F.I., target string variable	Stores status message of named file in specified string variable.
CALL "FMVALS", target numeric variable, target string variable	Sends the number of the current device to the numeric variable specified and the name of the current library to the string variable specified.
CALL "FORMAT", device address, volume I.D., 1, 1, owner I.D., master password, 1st, 2nd, 3rd, 4th, and 5th level chains	Records an information label on a new disc after searching for bad sectors. The disc is then automatically mounted. Any previous data is now inaccessible.

CALL "HERRS", device address, number of retries in last I/O, number of accumulated retries, number of successful I/O recoveries, number of unsuccessful I/O operations	Generates a count of I/O errors.
INIT	Resets all 4051 string, numeric, and array variables to an undefined state and closes all files.
INPUT #Ifn [,record number]: target variables [,target variables] ...	Inputs ASCII file data from the disc to the 4051; sequential or random access is permitted.
KILL F.I.[,master password]	Deletes existing files.
CALL "MOUNT", device address, target string variable	Activates disc and returns disc status message.
CALL "MRKBBG", device address, volume I.D., master password, address of defective space	Marks defective blocks on the disc so they aren't used again.
CALL "NEXT", Ifn, target string variable	Closes current file and opens the next one.
OLD F.I. [,"ASCII"]	Works like 4051 keywords "OLD" and "BOLD" (see the 4051 Graphic System Reference Manual).
ON EOF (Ifn)	See 4051 Graphic System Reference Manual.
Open F.I.[,"G"]; Ifn, type of access (R,F,U), target string variable	Opens a file for I/O operation.
PRINT #Ifn [,record number]: data item, data item . . .	Transfers ASCII data from the 4051 to the disc or tape.

GENERAL OPERATION

READ #lfn [,record number]: target variables [,target variables] . . .	Transfers binary data from the open file to the 4051.
CALL "RENAME", device address, old F.I., new F.I.	Renames files and changes passwords.
CALL "REWIND", lfn	Sets logical file pointer back to beginning of file.
SAVE F.I.["ASCII"]; line number [beginning, ending line number]	Stores program into disc file. A file will be created if none exists.
SECRET	Prevents future program listing.
CALL "SETTIM", date and time	Sets and starts the system clock.
CALL "SPACE", lfn, requested file size, target numeric variable, target numeric variable	Decreases the size of a file to the minimum size required for its contents.
CALL "TIME", target string variable	Stores the current time from the system clock in a specified string variable.
TYPE (lfn)	Works the same as TYP(0) does for internal magnetic tape.
UNIT device address	Selects current device address for all commands that do not contain a field for a device address.
CALL "UNIT", device address	Same as UNIT but allows a variable in the device address field.
CALL "USERLIB", partial F.I.	Specifies current library. The default current library is "SCRATCHLIB".
WRITE #lfn [,record number]: data item[,data item]	Transfers binary data from the 4051 to an open file.

Sample I/O Program

The program listed below 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 B). 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$)>0 THEN 170
140 PRINT "ENTER DATE AND TIME (DD-MOM-YY HH:MM:SS):";
150 INPUT A$
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 "]]THIS IS A SAMPLE PROGRAM FOR DEVICE ";D(I);"]]"
240 CALL "UNIT",D(I)
250 CALL "DRES",D(I)
260 CALL "FORMAT",D(I),"SAMPLE",1,1,"OWNER","PASS",3,3,3,3,3
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 #2,1:"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

```

GENERAL OPERATION

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 }
260 } Format disc
270 }

280 Create an ASCII, sequential file
290 Create a binary, random file

300 }
310 } Open each file and store message
320 }
330 }

340 Rewind sequential file

350 }
360 } Access files and display messages
370 }
380 }

390 Close both files
400
410 End program

Sample Program Output

```

RUN
ENTER DATE AND TIME (DD-MOM-YY HH:MM:SS):12-DEC-77 00:30:30
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)

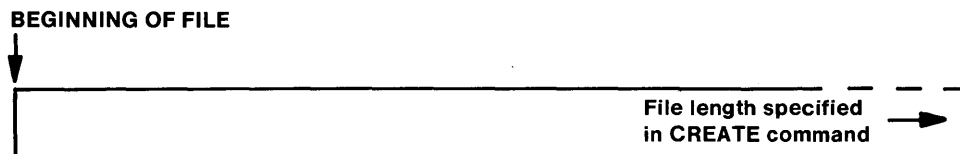
```

WHAT ARE 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:

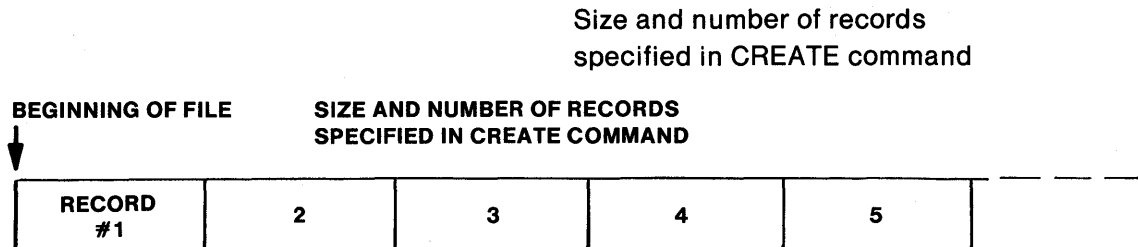


Information is entered sequentially. This means it must start at the first empty space and work towards the end of allocated space. The first empty space may be at the beginning of the file or just past the last data entered. Information retrieval also must be sequential and must start at the beginning 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 4051 mag tapes and discs.

GENERAL OPERATION

Random Files

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



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 4051 internal mag tape.

NOTE

Files are specified random or sequential in the CREATE statement.

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 each 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 if the file has been defined to allow this.

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 included 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 the file identifier can simplify 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.

GENERAL OPERATION

“Group” Open/Next File

Many times it is necessary to access several files consecutively. Normally, this would require an individual 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 placed 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 mag 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.

Simultaneous File Use

Once the system is notified which drive is to be accessed, up to 9 files may be opened and used. See OPENING MULTIPLE FILES in the OPEN command description.

Commands involved:

```
UNIT  
CALL “UNIT”
```

SECTION 3

CONTENTS

	Page
Introduction	3-1
What Is a Storage Structure	3-1

Section 3

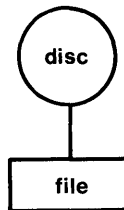
STORAGE STRUCTURE

INTRODUCTION

This section describes the concepts of file storage structures. It is necessary to understand these concepts to write file identifiers (F.I.s) which are necessary parts 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 being formed. To illustrate what "storage structure" is and why it is a useful concept, we will start by showing a "storage structure" containing a single file:

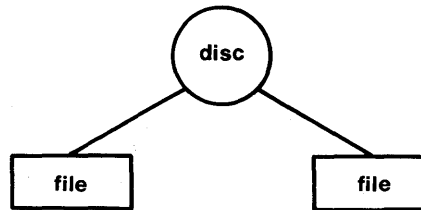


2380-14

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

STORAGE STRUCTURE

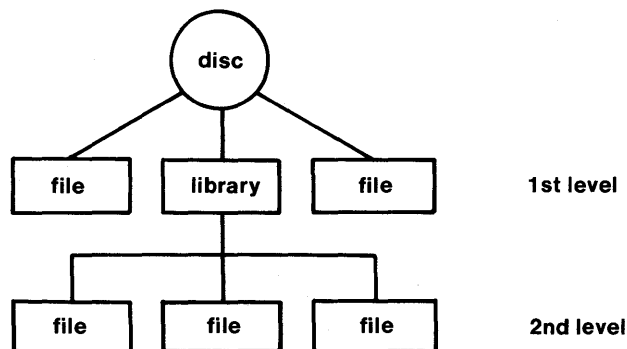
If another file were added, the structure would look like this:



2380-15

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

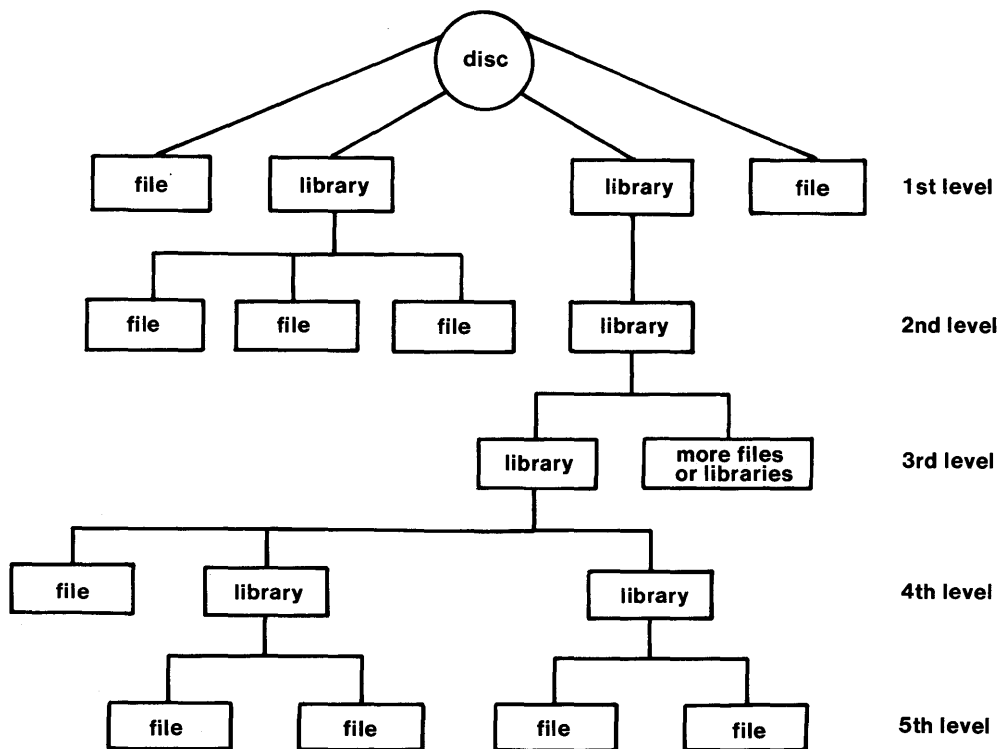
So far we've shown two files stored on the "1st level" of five storage levels provided by the system. If several files sharing a common characteristic or belonging to a particular user are to be added, they may be grouped under a "LIBRARY." If a library is used, the files will be placed on the next level; for example, three files grouped under a 1st level library will be placed on the 2nd level as shown below:



2380-16

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

A library is a “heading” and can only be used to lead to other libraries or files on lower levels. A library can never be used to hold data or programs, nor can a file ever lead to 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.



2380-17

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

Remember:

1. 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.
2. Files never have subsequent files or libraries.
3. There is no limit (other than space restrictions) to the number of libraries or files on a particular level.
4. Storage structures are limited to five levels.

SECTION 4

CONTENTS

	Page
Introduction	4-1
Keyword	4-1
Address	4-1
Argument	4-2
Syntax	4-3
Constant	4-3
String Constant	4-3
Numeric Expression	4-3
Numeric Variable	4-4
String	4-4
String Variable	4-4
Target String Variable	4-4
Target Numeric Variable	4-5
Command Delimiters, Punctuation and Spaces	4-5
Command Examples	4-6
How to Write a File Identifier	4-12
What is a File Identifier?	4-12
File Identifier Construction	4-15
Field Descriptions (Descriptive Form)	4-16
Use of First Level Libraries	4-17
Examples	4-18
Using Passwords in File Identifiers	4-22
How to Write a Password	4-22
How Much Access?	4-24
Extensions	4-25
File Identifier Delimiters	4-26
General Delimiter Information	4-27
Special Characters in File Identifiers	4-27
Special or Multiple File Selection (#)(*)(?)	4-27
Pound Sign (#)	4-28
Asterisk (*)	4-33
Question Mark (?)	4-37
USERLIB String Suppression (@)	4-38
Simplified SYSLIB Access (\$)	4-39

Section 4

HOW TO WRITE A COMMAND

INTRODUCTION

ROM pack commands, like other 4051 BASIC commands, must be written in a specific manner. Each of 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

All commands require keywords. The 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. Many keywords are preceded by the word CALL, which, while of no significance to the user, is required for system operation. The 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 to be active. The different types of addresses required are shown below.

device address This number, which must be an integer from 0 to 255, tells the system which device it will be communicating with. It may or may not be the same device specified in the UNIT command.

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.

HOW TO WRITE A COMMAND

lfn	Logical file number. The integer (from 1 to 9) entered here is specified in an earlier OPEN command and represents both the device address and the F.I. In some cases the lfn must be preceded by # (pound sign).
I/O address	The I/O address tells the system what device, other than the 4051 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", INIT, DELALL, 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. This means they must contain final information necessary for command execution. 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 4051 storage memory.

On the other hand, the variables in the argument may be used to supply information for a command execution rather than to store the results.

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

SYNTAX

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 number (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 shown with quotation marks ("DOCTORX") when entered in a command.¹ In some cases, a choice between a few single character constants is necessary. Those choices will be shown instead of the term "string constant" within the brackets like this:

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

Numeric Expression

The numeric requirement for the field may be represented by any of the following:

Numeric constant, such as 3

Numeric variable, such as M

Subscripted array, such as B(1),1

Logical or relational comparison enclosed in parenthesis, such as,

(A or B)

(A > B)

Valid combination of the above

¹When 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.

HOW TO WRITE A COMMAND

Numeric Variable

The entry may be any numeric variable allowed in regular 4051 operation. If a numeric variable is entered, it must represent the required number.

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 4051 operation (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 4051 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. Null strings ("") may be entered in the target string variable field to prevent file status messages from being stored.

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

Target Numeric Variable

This term is used when numeric information generated by the command is sent to the numeric variable entered in that field.

COMMAND DELIMITERS, PUNCTUATION AND SPACES

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

- | | |
|-----------------------------------|---|
| Commas, Colons and Semicolons ,;: | Act to separate fields. Their exact use is shown with individual command descriptions. |
| Quotation Marks " " | Serve to 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 in command construction. 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

HOW TO WRITE A COMMAND

The system has interpreted the E in E\$ as part of the keyword. When it moves to read the F.I. field, all it will see is \$ and an error condition will be created. 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 4051 Graphic System Reference Manual for further details.*

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.
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.
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 4051 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.

HOW TO WRITE A COMMAND

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\$.
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\$.

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 4051 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\$.

HOW TO WRITE A COMMAND

CALL "RENAME"	CAL"RENAME",1, "PERSONNEL/ MGRS/JONES", "PERSONNEL/ 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", "04- 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 4051 Graphic System Reference Manual.
UNIT	UNI 3	Specifies device 3 as the current device.

HOW TO WRITE A COMMAND

CALL "UNIT"	CAL"UNI",B	Specifies the device address in B as the current device.
CALL "USERLIB" (User Library)	CAL"USERLIB", "PORTLAND/ 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 "PORTLAND/INDUSTRY."
WRITE	WRI#3,40:A\$	Writes the binary string in A\$ to record 40 in lfn 3.

HOW TO WRITE A FILE IDENTIFIER

What is a File Identifier?

In STORAGE STRUCTURE (Section 3), we described libraries and files. Now we are going to discuss how a file identifier, or F.I., is used to create and access those libraries and files.

An F.I. is a collection of one to five names that tells the system:

the name and location of a file to be created

or

the name and location of an existing file

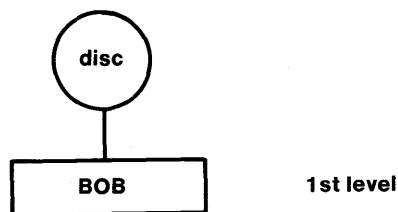
Whether the F.I. is used to create a new file or locate an existing file depends on the kind of command in which it is placed. The F.I. is useless by itself. It² must go inside a command.

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

Each F.I. contains one to five names³ that represent storage levels. For example, if you execute a command containing an F.I. of only one name like this:

“BOB”

then you can create or locate a storage structure consisting of one 1st level file, as shown in Figure 4-1:



2380-18

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

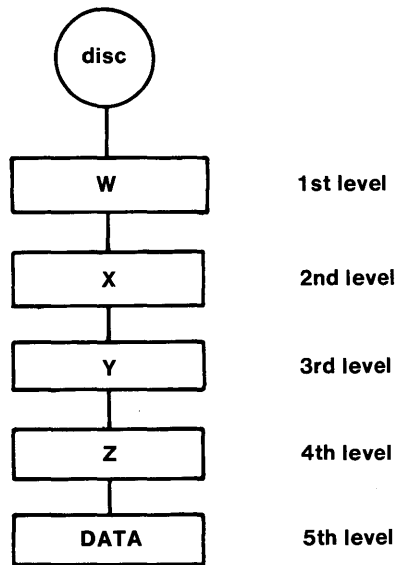
² Or a logical file number representing the F.I.

³ Exclusive of passwords and extensions described later.

If you enter an F.I. of five names like this:

“W/X/Y/Z/DATA”

then you can create or locate a storage structure consisting of one 5th level file and four libraries on the preceding four levels as shown in Figure 4-2.



2380-19

Figure 4-2. Sample Storage Structure (5 Levels, 1 File).

HOW TO WRITE A COMMAND

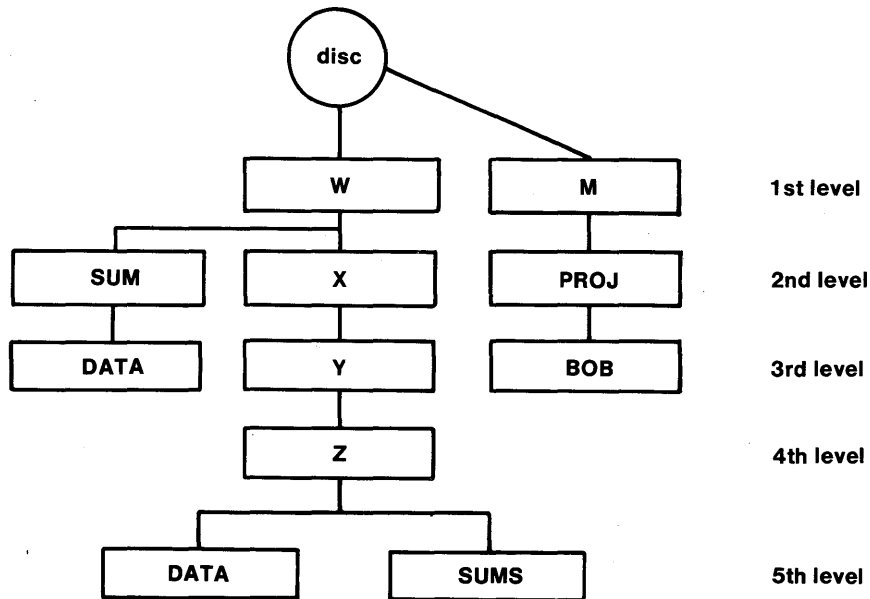
More files may be created or located anywhere on any existing storage structure by entering additional F.I.'s. For example, by executing:

"W/SUM/DATA"

"M/PROJ/BOB"

"W/X/Y/Z/SUMS"

in individual CREATE commands, the libraries SUM, M, PROJ and the files DATA, BOB, and SUMS will be added to the existing storage structure as shown in Figure 4-3.



2380-20

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

HOW TO WRITE A COMMAND

Field Descriptions (Descriptive Form)

@(The Commercial "at" Sign)

The "@" is used to circumvent the "current library." See USERLIB String Suppression in this section and the CALL "USERLIB" command description in Section 5.

1. Name of Library or File

The entry here specifies the name of the 1st level library or file. For convenience, these libraries fall into groups named SYSLIB, USERLIB, or SCRATCHLIB. Each group is accessed differently. See USE OF 1ST LEVEL LIBRARIES following these field descriptions.

2. Name of Library or File

The entry here specifies the name of the 2nd level library or file. The name may be 1 to 10 characters long. The first character must be alphabetic, the rest alphanumeric.

3. Name of Library or File

The entry here specifies the name of the 3rd level library or file. The name may be 1 to 10 characters long. The first character must be alphabetic, the rest alphanumeric.

4. Name of Library or File

The entry here specifies the name of the 4th level library or file. The name may be 1 to 10 characters long. The first character must be alphabetic, the rest alphanumeric.

5. Name of File

The entry here specifies the name of the 5th level file. Since this is the "lowest level," all entries here must be files. The character requirements are the same as for the above fields.

USE OF 1ST LEVEL LIBRARIES

There are three types of 1st level libraries:

SYSTEM LIBRARY
(SYSLIB)

USER LIBRARY
(USERLIB)

SCRATCH LIBRARY
(SCRATCHLIB)

SYSLIB: This library may be used to contain 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 may 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.

SCRATCHLIB: This library may contain 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 or default library.

USERLIB: Enter "@" then the name of the user library then the rest of the library and filenames.
Example: "@MYLIBRY/A/B/C". This will suppress the current or default library.

SCRATCHLIB: Leave out the name of the 1st level library, SCRATCHLIB, then enter the rest of the library and filenames.
Example: "A/B/C".

HOW TO WRITE A COMMAND

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:

`"@SCRATCH/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.⁴ To see how characters are used as delimiters in F.I. construction, see FILE IDENTIFIER DELIMITERS in this section.

SYSLIB

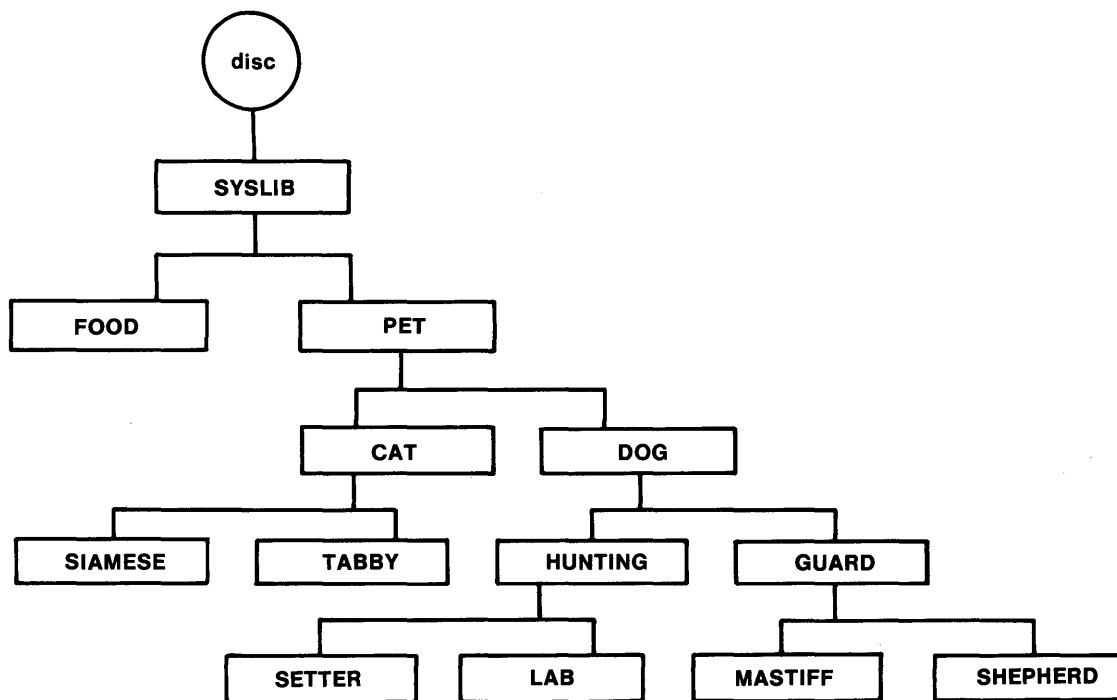
Examples 1 through 3 show how F.I.'s should be written for the "SYSLIB" storage structure (Figure 4-5). These F.I.'s circumvent any current library. This means the system will not look in the current library for the file.

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.

⁴"Storage structure" means all libraries and files under these 1st level libraries.



2380-22

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

Example 2:

“\$FOOD”

This example locates the file “FOOD” in “SYSLIB.”

Example 3:

A\$

If the statement

A\$ = “\$FOOD”

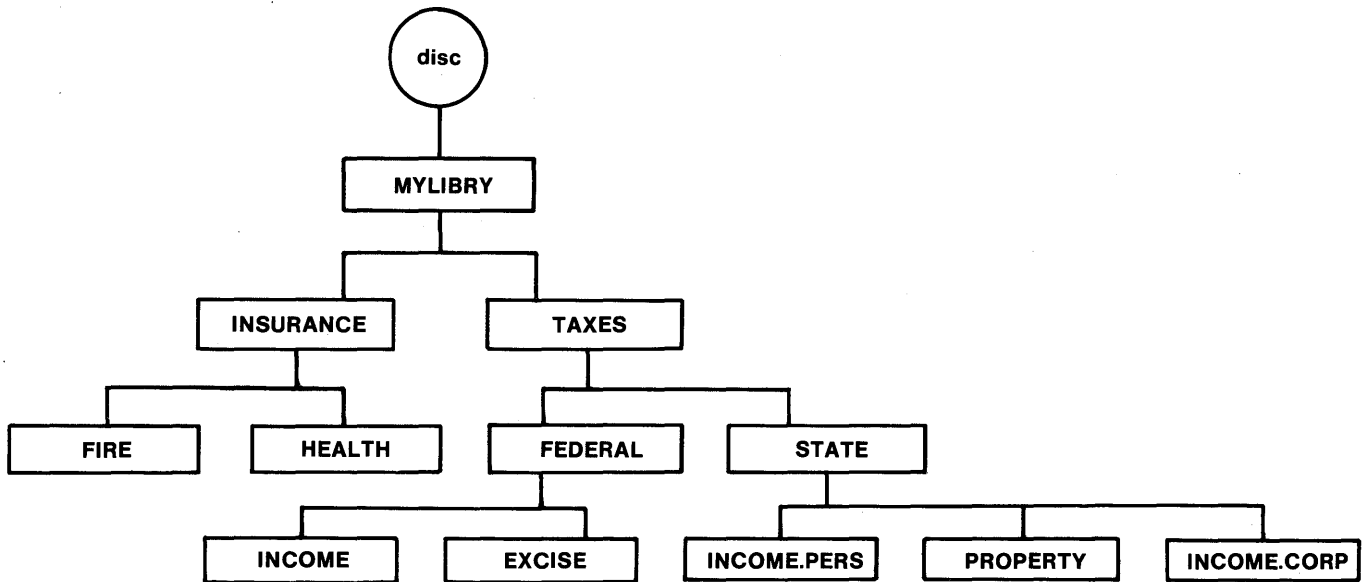
is executed previously, the same file as in example 2 will be located if A\$ is entered in the F.I. field. Although this file is on the 2nd level, it has no subsequent files and is, therefore, a “lowest level” file.

HOW TO WRITE A COMMAND

USERLIB

The following examples show how F.I.'s should be written for the "MYLIBRY" storage structure in Figure 4-6.

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.



2380-23

Figure 4-6. Sample Storage Structure (4 Levels, 7 Files).

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:

"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\$

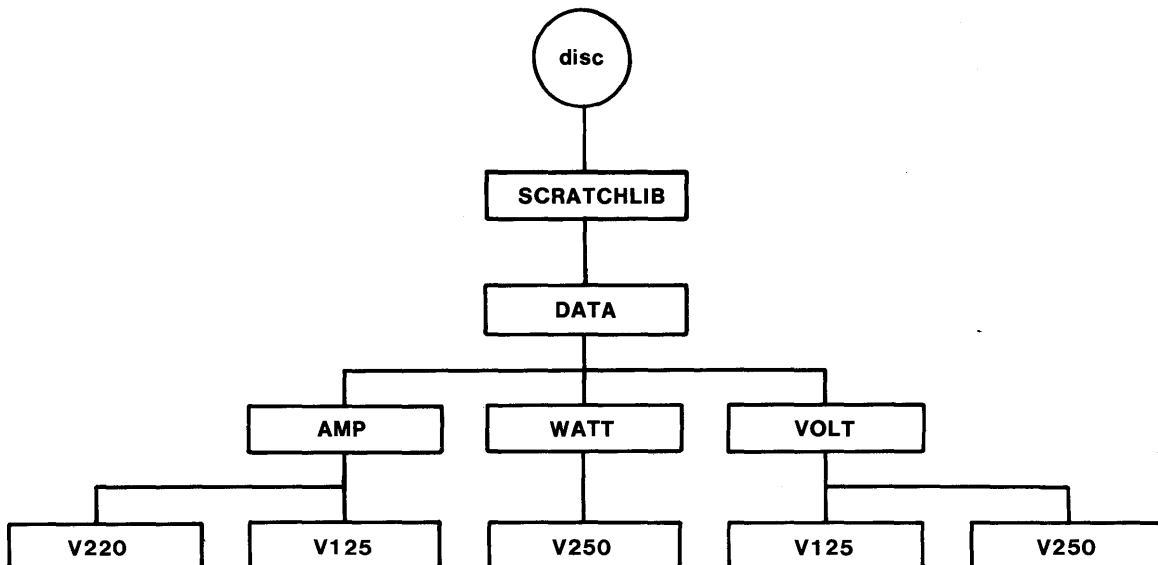
If the statement

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 4-7.



2380-24

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

HOW TO WRITE A COMMAND

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 default library.

Using Passwords in File Identifiers

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.⁵

How to Write a Password

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.

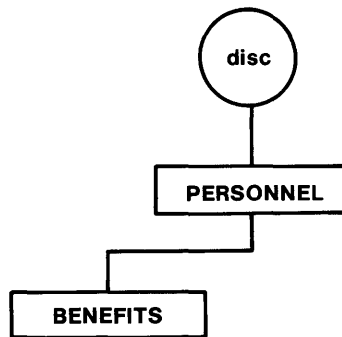
⁵Unless a master password is used. See CALL "FORMAT" command description in Section 5.

EXAMPLE:

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"

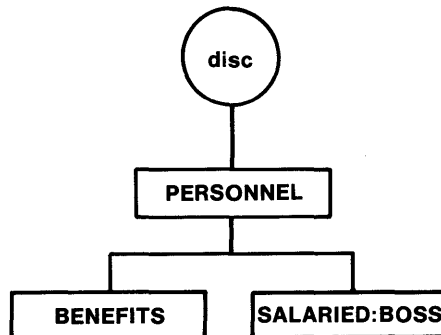
If the existing structure looks like this:



2380-25

Figure 4-8. Sample Storage Structure (2 Levels, 1 File).

the F.I. will result in the revised storage structure shown in Figure 4-9.



2380-26

Figure 4-9. Sample Storage Structure (2 Levels, 2 Files, 1 With Password).

HOW TO WRITE A COMMAND

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.

How Much Access?

Specifying a *library* password prevents any future access to subsequent files unless that password is used.

The degree of access to a *file*, however, is dependent not only on the use of a password but also on the attribute assigned to the file in the CREATE command. An "R" must be assigned to a file if it is to be private. A "U" must be assigned if the file is to be public.

The chart below shows how public and private file status is affected by use or nonuse of an assigned password.

	When "U" (public) is specified as an attribute, these commands may be executed.	When "R" (private) is specified as an attribute, these commands may be executed.
When specified password is supplied:	READ OLD WRITE SAVE PRINT KILL INPUT RENAME APPEND COPY... TO	READ OLD WRITE SAVE PRINT KILL INPUT RENAME APPEND COPY... TO
When specified password is not supplied:	READ INPUT APPEND OLD	No access allowed

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 4-10 and the description following it show how extensions are used:

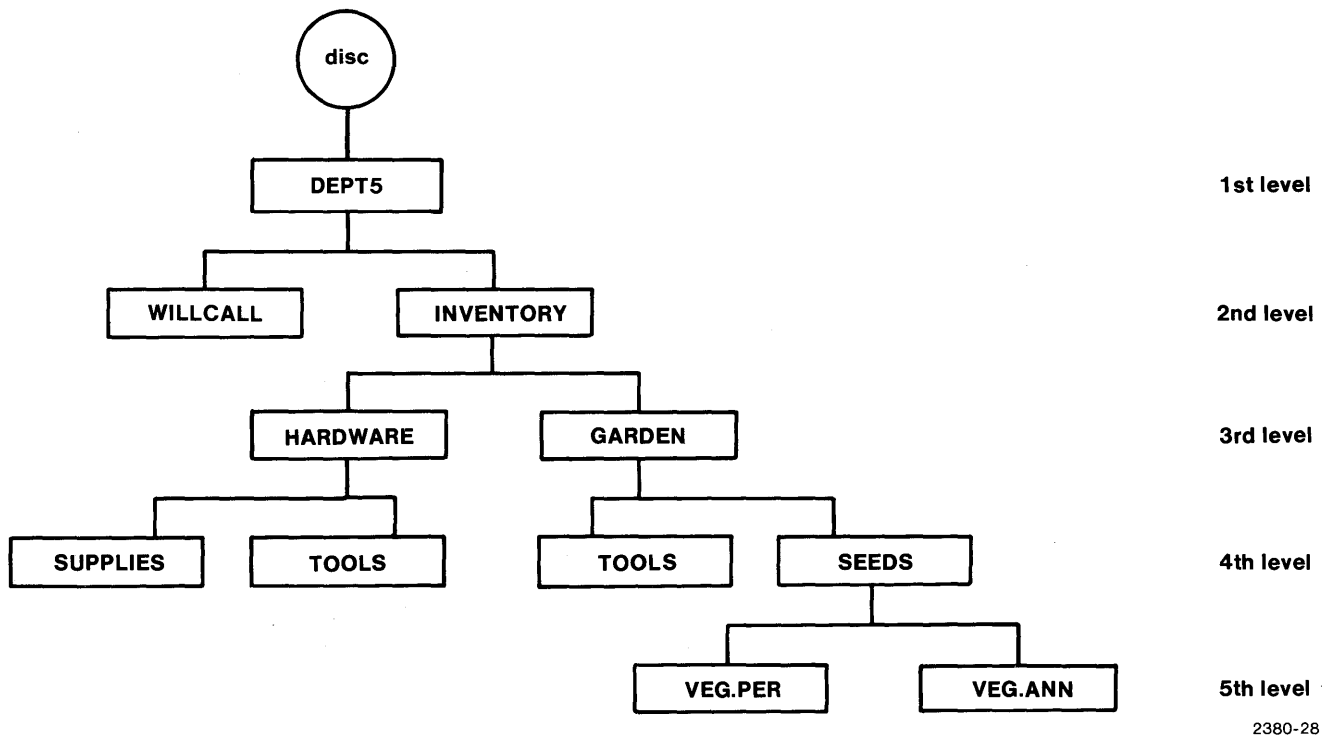


Figure 4-10. Sample Storage Structure (5 Levels, 6 Files).

HOW TO WRITE A COMMAND

This structure shown in Figure 4-10 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 "PER" and "ANN" were added onto the "VEG" files. This is to allow the 4051 to distinguish between them when only one is to be accessed. You may have noticed that no extensions were specified for the files named "TOOLS". This is because they are under different libraries. Once the 4051 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.

File Identifier 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.

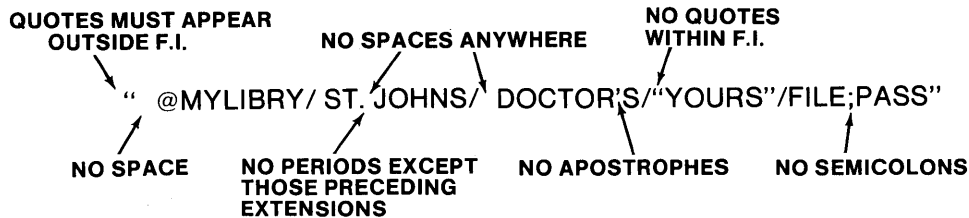
EXAMPLE:

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

General Delimiter Information

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

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



SPECIAL CHARACTERS IN FILE IDENTIFIERS

In HOW TO WRITE A FILE IDENTIFIER we discussed exactly how F.I.'s must be written. That information described how to write an F.I. for a single file. An F.I., however, may be far more useful with the addition of 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 order of their hashing address. 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.

HOW TO WRITE A COMMAND

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:

1. Only one “#” may be used in any F.I.
2. Slash marks (/) cannot be used immediately adjacent to a “#”. The exception is the slash marks assigned automatically in the CALL “USERLIB” command since the slash mark in this instance is “invisible” to the “#.”
3. Only one file level (with an optional extension) may be specified to the right of a “#.”

EXAMPLES:

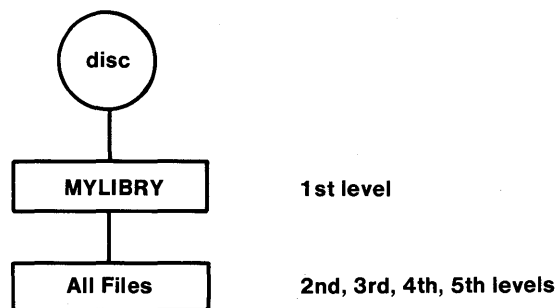
These examples show the use of the “#” in an F.I.

Assume “MYLIBRY” is the current library.

Example 1:

“#”

This example locates all files in “MYLIBRY” as shown in Figure 4-11.



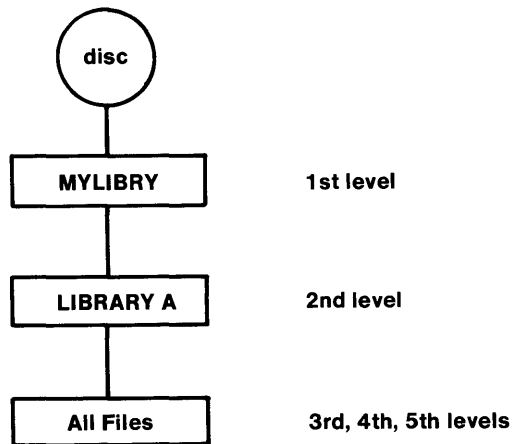
2380-27

Figure 4-11. 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 4-12.



2380-48

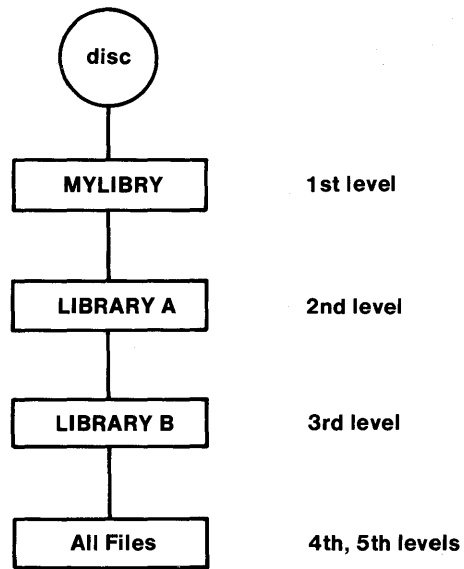
Figure 4-12. Sample Storage Structure (3 Levels).

HOW TO WRITE A COMMAND

Example 3:

"A/B#"

This example locates all files on the 4th and 5th levels below library "B" as shown in Figure 4-13.



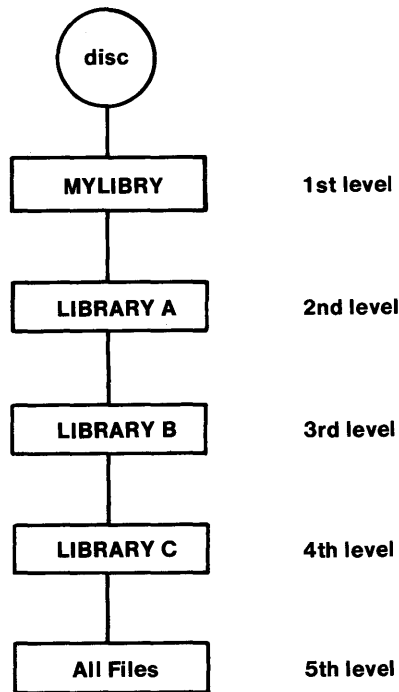
2380-29

Figure 4-13. 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 4-14.



2380-30

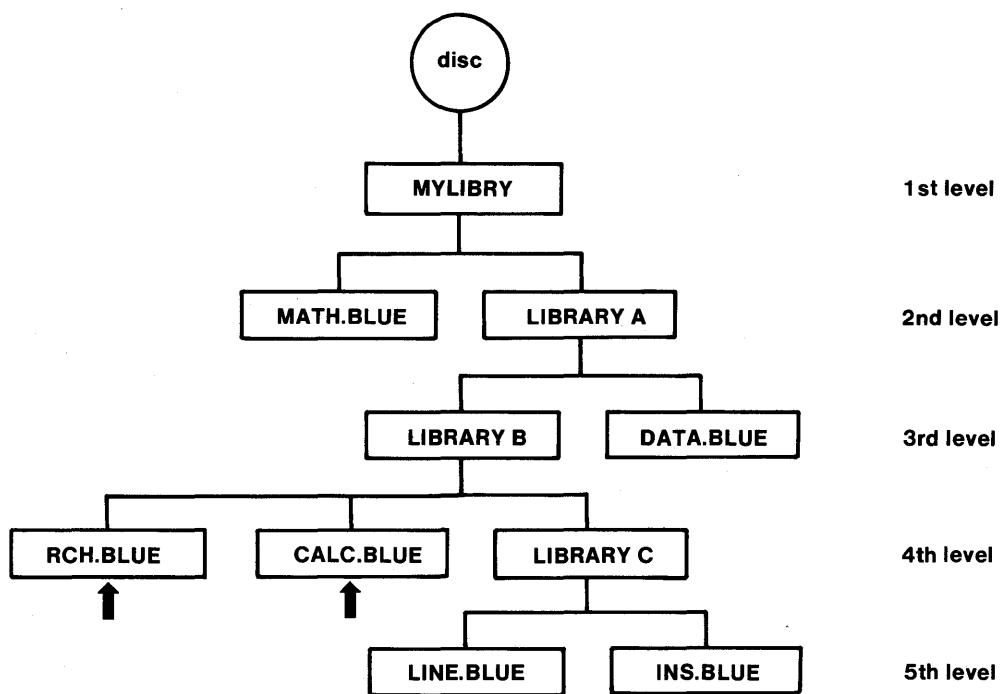
Figure 4-14. Sample Storage Structure (5 Levels).

HOW TO WRITE A COMMAND

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.



2380-31

Figure 4-15. 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:

1. 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.
2. An asterisk cannot be used to locate a file or library with a password.
3. Characters in a name cannot be used to the right of an asterisk; for example, "DO*" is legal but "D*N" is not.

EXAMPLES:

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.

HOW TO WRITE A COMMAND

Example 1:

`**/B/C`

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

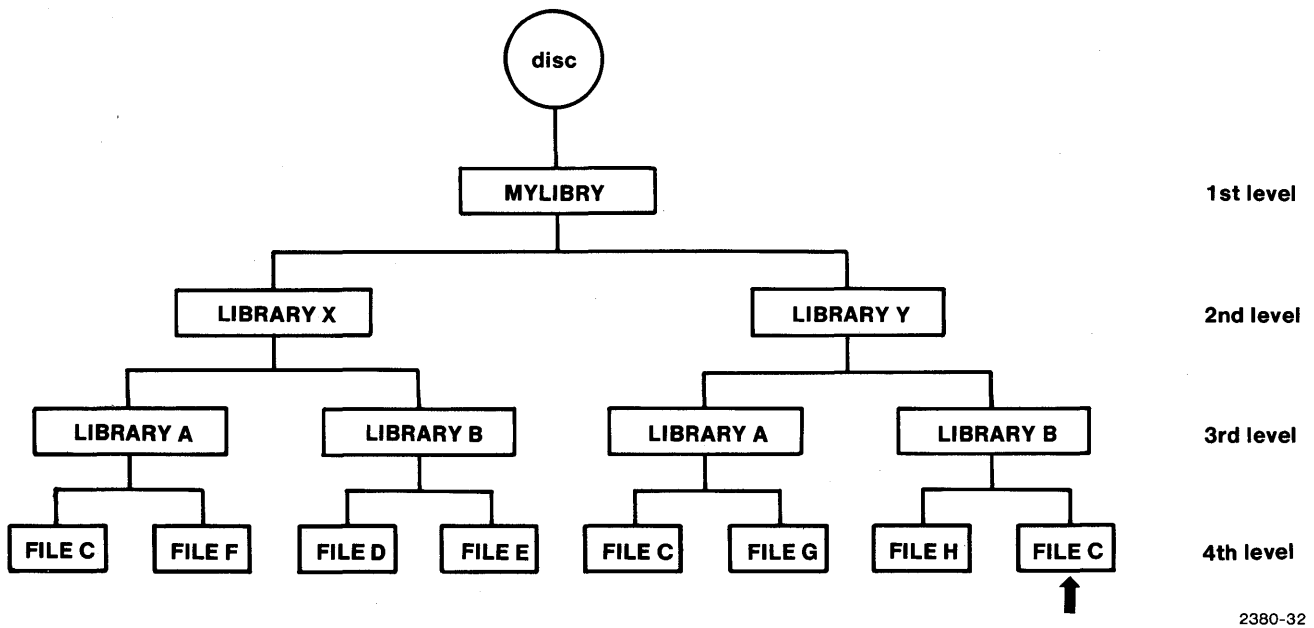


Figure 4-16. Sample Storage Structure (4 Levels, 8 Files).

Example 2:

"A/*/C"

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

Example 3:

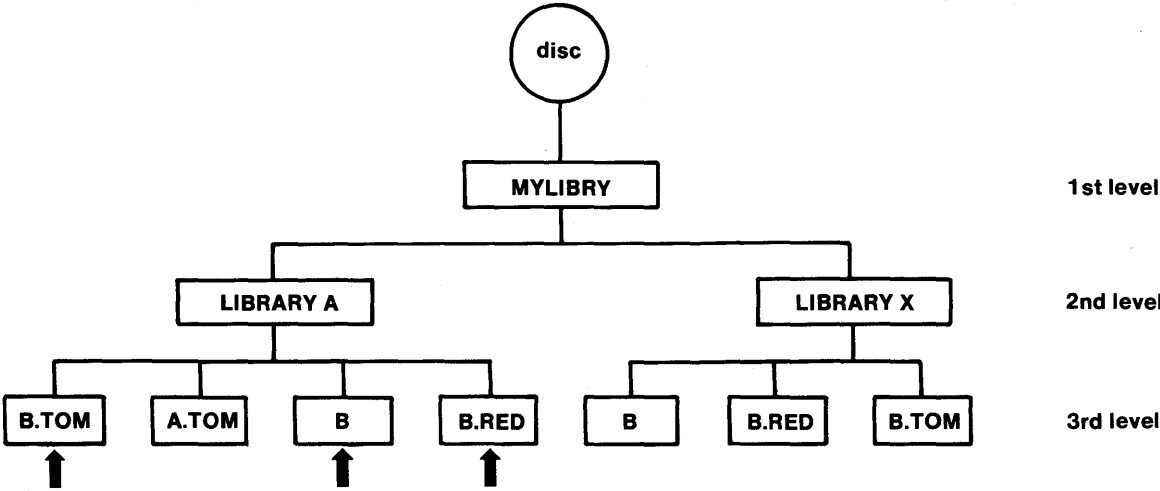
"A/B/*"

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

Example 4:

"A/B.*"

This example locates all 3rd level files "B" with any extension under 2nd level library "A." The arrows in Figure 4-17 show these files.



2380-33

Figure 4-17. Sample Storage Structure (3 Levels, 7 Files).

HOW TO WRITE A COMMAND

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.

Limitations:

The question mark cannot represent more than a single character.

EXAMPLES:

These 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.

HOW TO WRITE A COMMAND

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 part of an F.I. is entered with a CALL "USERLIB" command, that portion is accessed and 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".

When line 300 is executed, the system automatically accesses the current library, "YOURLIBRY," and locates 1st 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 default current library is 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. 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 4051) 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

Introduction	5-1
How to Read Command Descriptions	5-1
Format	5-2
Purpose	5-2
Syntax Form	5-2
Descriptive Form	5-3
Field Definitions	5-3
General Information	5-3
Prerequisites	5-3
Examples	5-3
System Commands	5-5
Controller/Device Commands	5-19
File Management Commands	5-43
File I/O Commands	5-89

Section 5

COMMAND DESCRIPTIONS

INTRODUCTION

This section explains how to read command descriptions. It then describes all commands unique to the 4907 File Manager.

HOW TO READ COMMAND DESCRIPTIONS

4907 File Manager commands have been grouped into the following four categories:

1. SYSTEM COMMANDS
2. CONTROLLER/DEVICE COMMANDS
3. FILE MANAGEMENT COMMANDS
4. FILE I/O COMMANDS

The following is a brief definition of each category. Read through it before proceeding the individual commands.

1. SYSTEM COMMANDS

These commands affect the entire 4907 File Manager system. They include commands that set the system clock, the current device, the current library and more.

2. CONTROLLER/DEVICE COMMANDS

These commands activate and deactivate the drive, format the disc, etc. These are basically preparatory commands, and, therefore, affect file management and I/O only indirectly.

3. FILE MANAGEMENT COMMANDS

These commands create, rename, enlarge, reduce and reserve files as well as carry out many other file activities.

4. FILE I/O COMMANDS

These commands are used to place information into or retrieve information from files.

COMMAND DESCRIPTIONS

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, HOW LARGE SHOULD FILES OR FILE RECORDS BE? is included in the CREATE command description.

Purpose

This section explains the function of the command.

Syntax Form

The syntax form shows the number of fields, delimiters and punctuation required. However, for simplicity, the required quotation marks are generally not shown in the syntax form (except for keywords which do show quotes). 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 and braces are shown for the following reasons:

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 there is a choice between the terms, characters, classes or numbers within the braces.

Braces and brackets are shown in descriptive form for the same reasons.

Descriptive Form

After the complete keyword, this form provides a one or two word definition of the purpose of each field.

Field Definitions

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

General Information

Operational peculiarities for 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 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, or statements in programs is demonstrated in INPUT and READ command descriptions as well as SAMPLE PROGRAMS (Appendix F), USING PRINT AND INPUT IN A PROGRAM (Appendix G), and USING WRITE AND READ IN A PROGRAM (Appendix H).

NOTE

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

SYSTEM COMMANDS

These commands affect the entire 4907 File Manager system. They include commands that set the system clock, the current device, the current library and more.

DELALL (Delete All)	5-5
CALL "FMVALS" (File Manager Values)	5-6
INIT (Initialize)	5-8
CALL "SETTIM" (Set Time)	5-9
CALL "TIME"	5-11
UNIT	5-12
CALL "UNIT"	5-14
CALL "USERLIB"	5-16

DELALL (Delete All)**Purpose**

This command erases everything currently in the 4051 memory.

This is a 4051 BASIC command. It is included here because its execution has a direct effect on system operation.

SYNTAX FORM

DEL ALL

DESCRIPTIVE FORM

DELETE ALL

General Information

Execution of this command closes all disc files, sets the current device to zero (0), and specifies "SCRATCHLIB" as the current library.

Prerequisites

None.

CALL "FMVALS"

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$
```

INIT

INIT (Initialize)

Purpose

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

This is a 4051 BASIC command. It is included here because it has a direct effect on 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.

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

CALL "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-MOM-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.

The syntax for this range is 0:0:0 to 23:59:59.

CALL "SETTIM"

General Information

NOTE

The system clock must be operating whenever the system is being used. When the clock has been set, the clock indicator on the controller front panel is out. The system clock must be restarted 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 December 32 on non-leap years, but makes no provision for February 29 on leap years.

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 "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\$.

UNIT

UNIT

Purpose

The UNIT command specifies the current device. Commands containing F.I.'s or lfn's but no 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.

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 technician 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 lfn's but no device addresses attempt to carry out their operation on 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 address 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 FILE IDENTIFIERS 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. This automatically specifies SCRATCHLIB as the current library.

Prerequisites

None.

Examples

Example 1:

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".

COMMAND DESCRIPTIONS

CALL "USERLIB"

Example 2:

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"

CONTROLLER/DEVICE COMMANDS

These commands activate and deactivate the drive, format the disc, etc. These are basically preparatory commands and therefore, affect file management and I/O only indirectly.

CALL "COMPRS" (Compress)	5-19
CALL "CUSTAT" (Controller Unit Status).....	5-12
CALL "DISMOUNT"	5-23
CALL "DREL" (Device Release)	5-25
CALL "DRES" (Device Reserve)	5-27
CALL "DSTAT" (Device Status).....	5-29
CALL "FFRMT" (Fast Format)	5-31
CALL "FORMAT"	5-32
CALL "HERRS" (Error Status)	5-36
CALL "MOUNT"	5-38
CALL "MRKBBG" (Mark Bad Block Group).....	5-40

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 created. To collect and group 100% of the free space, a CALL "DUP" command must be executed.

COMMAND DESCRIPTIONS

CALL "COMPRS"

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).

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.

COMMAND DESCRIPTIONS

CALL "CUSTAT"

General Information

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

Prerequisites

Set clock.

Example

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

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

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.

COMMAND DESCRIPTIONS

CALL "DISMOUNT"

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

CALL "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.

COMMAND DESCRIPTIONS

CALL "DREL"

Prerequisites

Device must be reserved.

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

CALL "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.

COMMAND DESCRIPTIONS

CALL "DRES"

Prerequisites

All files must be closed and clock 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 being addressed. This includes the disc identification number, name of disc, etc. 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 to hold more than 72 characters; i.e., DIM A\$ (N). 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.

COMMAND DESCRIPTIONS
CALL "DSTAT"

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 to the 4051 screen.

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; however, the keywords CALL "FFRMT" replace CALL "FORMAT".

General Information

Attempting a CALL "FFRMT" on a volume not previously formatted or a volume 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 old disc with a CALL "FFRMT" command may prevent full use of the new disc.

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

Prerequisites

Device must be reserved.

CALL "FORMAT"

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.
- Conduct a surface analysis to detect 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".

<p>SYNTAX FORM</p> <p>CAL "FORMAT", numeric expression, string, numeric expression, numeric expression, string, string, numeric expression, numeric expression, numeric expression, numeric expression, numeric expression.</p> <p>DESCRIPTIVE FORM</p> <p>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.</p>

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.
volume number	Enter 1.
number of volumes	Enter 1.

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.

master password

The master password may 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 entry may be up to 10 characters. The first character must be alphabetic; the rest may be alphanumeric.

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

The disc controller may be requested to speed up its file location processes; however, more space on the disc must be devoted to implementing this increase in speed. This means less room is available for file storage. The entries in these five fields may be used to vary this speed/space ratio when executing such search commands as DIRECTORY, RENAME, COPY . . . TO, OPEN (GROUP) and CALL "FILE".

The entries in each field may be 1 through 10. The higher the number, the faster the search commands are executed. Approximately 5K bytes of extra space are consumed if 10's instead of 1's are entered. The following entries are suggested for the five fields:

1st level = 7
2nd level = 7
3rd level = 3
4th level = 3
5th level = 3

CALL "FORMAT"

If more speed is required, the numbers may be increased.

If more space is required the numbers may be decreased.

General Information

Whenever a volume (disc) is formatted all, existing data is erased. To be sure the user 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?

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",7,7,3,3,3
```

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 specified as 7,7,3,3 and 3, 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"

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 around 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 up 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 up 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 up 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 as 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.

Examples

See 4907 Service Manual.

CALL "MOUNT"

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	<p>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.</p> <p>If a fast command execution is necessary, the message may be eliminated by entering two quotation marks ("") in this field. See DEVICE AND FILE STATUS MESSAGES (Appendix A) for details on message content.</p>

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\$.

Example 2:

```
150 CALL "MOUNT",B,""
```

This example tells the system that the disc with its address in B is ready for use. There is no device status message because the last field contains a null string ("") rather than a string variable.

CALL "MRKBBG"

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. |
| 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. |

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 the specific cause of the error. These last two digits indicate whether the message is the result of mechanical or bad block problems. See General Information below.

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 bad file to another location.
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 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.

COMMAND DESCRIPTIONS

CALL "MRKBBG"

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.

FILE MANAGEMENT COMMANDS

These commands create, rename, enlarge, reduce, open and close files as well as carry out many other file activities.

ASSIGN.....	5-43
CLOSE.....	5-45
COPY ... TO	5-47
CREATE	5-54
DIRECTORY.....	5-59
CALL "DUP" (Duplicate)	5-62
END	5-64
CALL "FILE"	5-65
KILL	5-67
CALL "NEXT"	5-69
ON EOF (On End-of-File).....	5-71
OPEN.....	5-73
CALL "RENAME"	5-78
CALL "REWIND".....	5-81
SECRET	5-83
CALL "SPACE".....	5-84
TYP (TYPE)	5-86

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.

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.

ASSIGN

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 4051 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 DELALL commands close all open files. Pressing the BREAK key twice also closes all open files.

COMMAND DESCRIPTIONS

CLOSE

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.

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. #1 ,source device address TO F.I. #2 ,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. #1

This entry must match the F.I. of the file to be copied. Do not enter the "#1." 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.

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. #2

The entry here shows not only where the file is to be copied, but also name changes to be made. Remember, two files with the same F.I. cannot be duplicated on the same disc. Do not enter the "#2."

COMMAND DESCRIPTIONS

COPY . . . TO

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. #2 already exists on the second device.
- The specified library contains no files.
- The device (or devices) 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 library,2TO"@MYLIBRY/DOG/CAT library",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 COPY$A,TOB$,B
```

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

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 F.I. #2. 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 "#." Don't confuse the F.I. #2 which means "File Identifier number two" and the special character pound sign (#).

Multiple File Transfers. Using "#", "*" or the "?" (see SPECIAL CHARACTERS IN FILE IDENTIFIERS 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.

COPY... TO**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".

COMMAND DESCRIPTIONS

COPY... TO

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 only 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

CREATE

Purpose

The CREATE command creates file space on the disc, assigns the attributes and F.I., including passwords and extensions, if any. The last two fields in this command are used to 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.
attributes	<p>If no entry is made here, the system defaults to the following conditions:</p> <p>B (BINARY): Only binary data may be entered.</p> <p>R (PRIVATE): Read operations may not be executed without using the password if a password has been specified.</p> <p>S (SCATTERED): Files will be located wherever there is space (on a single disc only).</p> <p>N (NOT COMPRESSIBLE): Files may not be compressed when file space exceeds current stored program or data.</p>

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 an 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.

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 HOW LARGE SHOULD THE FILE OR FILE RECORDS BE? 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 HOW LARGE SHOULD THE FILE OR FILE RECORDS BE? 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.

CREATE**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, an ASSIGN command may be executed to change the attribute "C" to "S."

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

Extended file space is always 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 entry in the last field 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.

How Large Should a File or File Record Be?

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 WRITE and PRINT.

ASCII	
Numeric values and strings	1 byte per character + 1 byte for EOR*
BINARY	
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

CREATE

See 4051 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.

DIRECTORY

Purpose

The DIRECTORY command generates file status messages which may be displayed on the 4051 screen or stored on the 4051 internal magnetic tape. If the 4051 is equipped with Option 10, status messages may be output to the printer.

SYNTAX FORM

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

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 to the 4051 screen. If the optional @33: is entered, the messages are stored on the 4051 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 the 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. (names) 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.

COMMAND DESCRIPTIONS
DIRECTORY

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.

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 DIRECTORY0,"@MYLIBRY #"
```

This example generates status messages for all eligible files in "MYLIBRY." Since there is no I/O address entry, all messages display to the 4051 screen. A 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 printed to the 4051 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 DIRECTORY
```

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,"@SCRATCHLIB#"
```

This example displays to the 4051 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 DIRECTORY
```

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 to the 4051 screen "name only" file status messages for all files contained in MYLIBRY.

CALL "DUP"

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.

<p>SYNTAX FORM</p> <p>CAL "DUP", numeric expression, numeric expression, numeric expression</p> <p>DESCRIPTIVE FORM</p> <p>CALL "DUP", source device address, target device address, compress control</p>

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

1. 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.
2. When the CALL "DUP" command is executed, ALL INFORMATION ON THE TARGET DEVICE IS ERASED.
3. All programs transmitted are duplicated as whole entities. No files are duplicated in "scattered" form.
4. This command is not completed if more than eight files are open.
5. 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
```

The last three entries here are actual values rather than numeric variables. 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

END

Purpose

END, a 4051 BASIC command, stops all program execution, closes all files and returns control to the 4051 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 "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 FILE IDENTIFIERS in Section 4. Passwords specified in the CREATE command are not necessary when writing the F.I. for this command.
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.

CALL "FILE"

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 not be completed 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\$.

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.
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.

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.

COMMAND DESCRIPTIONS

KILL

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 FILE IDENTIFIERS 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 "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.

CALL "NEXT"

General Information

1. If a CALL "NEXT" command is executed after a normal OPEN (not group) command, an error message is displayed.
2. 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.
3. If OPEN "G" locates no files, the first CALL "NEXT" generates an error message.
4. If a CALL "NEXT" has located no files the following CALL "NEXT" generates an error message.
5. When CALL "NEXT" is executed, the current file is closed.
6. 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\$.

ON EOF (On End-Of-File)

Purpose

This form of the 4051 BASIC ON . . . THEN statement allows continued program operation when the end-of-file is encountered. It is particularly useful when retrieving information from a 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

1. If no ON EOF command has been entered in the program, an error message is generated when the end-of-file has been reached.
2. If the 4051 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.
3. See the ON . . . THEN command description in the 4051 Graphic System Reference Manual for further details.

COMMAND DESCRIPTIONS
ON EOF

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:

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

SYNTAX FORM

OPEN 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 must match the current 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 FILE IDENTIFIERS 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.

COMMAND DESCRIPTIONS

OPEN

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).



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.

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, which may or may not exist (depending on how the address straps on the disc drives have been set).

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

Since it occasionally is necessary to conveniently review what has just been entered, the CALL REWIND command may be executed. 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.

Prerequisites

Disc must be mounted but device must not be reserved.

Examples

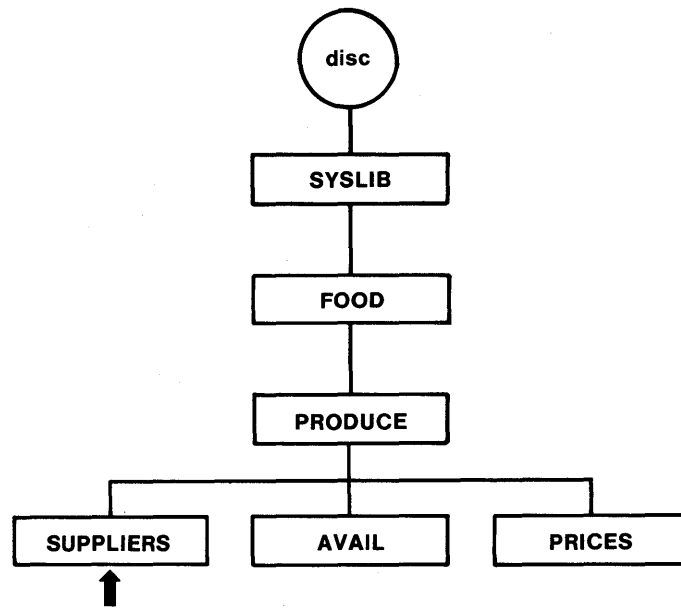
Example 1:

```
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 Fig. 5-1.

OPEN



2380-34

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

Example 2:

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

This example opens 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.

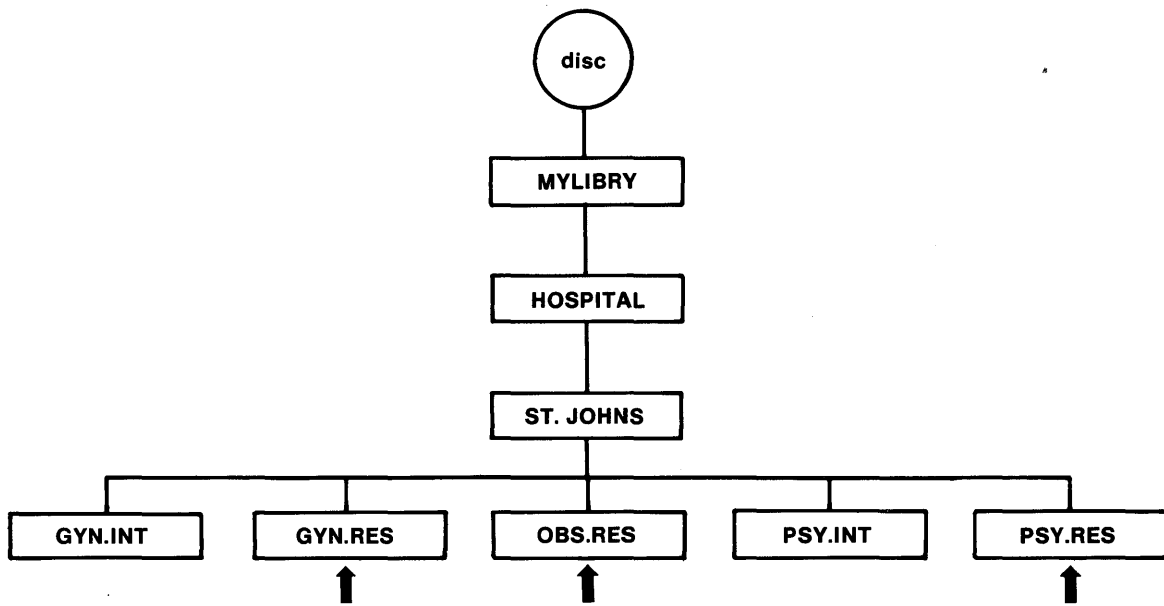
Example 3:

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 this is a "GROUP" OPEN, a "G" is specified in the next field. If anything other than a "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 Fig. 5-2.



2380-35

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)
```

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.

new F.I. The new F.I. is entered here.

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.

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:

1. The CALL "RENAME" command will not copy files to another device.
2. 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

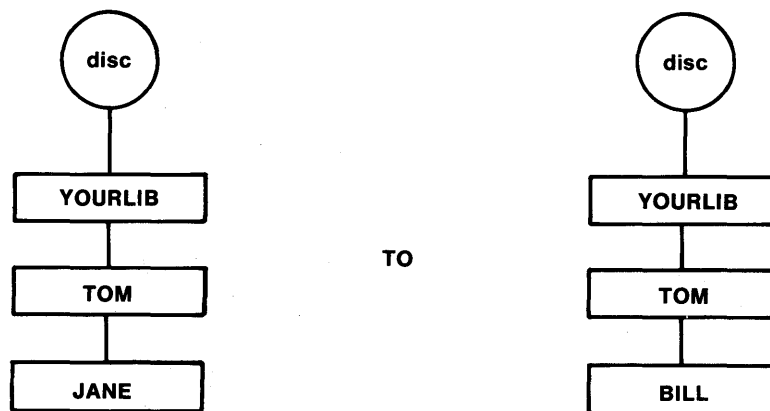
Disc must be mounted but device must not be reserved.

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."



2380-36

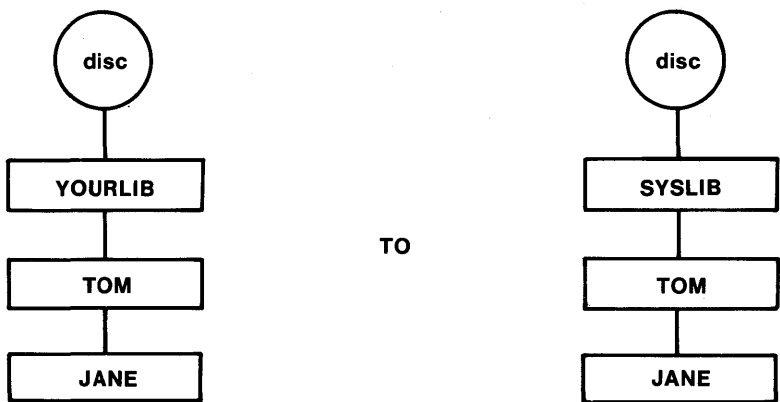
Figure 5-3. Sample Storage Structure Illustrating How to Change File Names.

CALL "RENAME"

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.



2380-37

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.

COMMAND DESCRIPTIONS

CALL "REWIND"

Example

400 CALL "REWIND",2

This example rewinds the pointer to the beginning of the file identified by lfn 2.

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 4051 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 "SPACE"

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

CALL "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.

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, not open, or 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 or TYP in the 4051 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 4051 Graphic System Reference Manual for further details.

Table 5-1

DATA TYPE TABLE

0	Empty File or File Not Open
1	End of File Character
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.

FILE I/O COMMANDS

These commands are used to place information into or to retrieve information from files.

- APPEND 5-89
- INPUT 5-92
- OLD 5-95
- PRINT..... 5-97
- READ 5-101
- SAVE 5-104
- WRITE 5-106

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.
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 no entry is made, the system defaults to a binary append.

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 the 4051 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 program being brought in from a file. Only those 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.

INPUT

INPUT

Purpose

The INPUT command reads ASCII data that has been stored in the designated file with a PRINT command. It operates like the 4051 BASIC INPUT command with the following exceptions:

1. 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.
2. 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.
3. The primary address character is "#".

SYNTAX FORM

INP # constant [,numeric expression] : $\left\{ \begin{array}{l} \text{array variable} \\ \text{string variable} \\ \text{numeric variable} \end{array} \right\} \left[\left\{ \begin{array}{l} \text{array variable} \\ \text{string variable} \\ \text{numeric variable} \end{array} \right\} \right] \dots$

DESCRIPTIVE FORM

INPUT # lfn [,record number] : target variables for incoming data $\left[\begin{array}{l} \text{target variables} \\ \text{for incoming data} \end{array} \right] \dots$

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

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

Prerequisites

File must be open.

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 checks for 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.

COMMAND DESCRIPTIONS

INPUT

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 lfn 3. The system then reads everything in that record up to the first CR 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.

OLD**Purpose**

The OLD command loads a BASIC program in either binary or ASCII format from the disc file into the 4051 memory.

SYNTAX FORM

OLD F.I. [,string]

DESCRIPTIVE FORM

OLD F.I. [,"ASCII"]

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.
ASCII	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

1. An error message appears if the program is not found or is a data file.
2. If the program was originally stored in binary, it must be retrieved in binary. The same applies to programs stored in ASCII.
3. 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.
4. 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.

COMMAND DESCRIPTIONS

OLD

Prerequisites

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

Example

140 OLD "@MYLIBRY/STOCKS"

This example transfers a binary program from the file "STOCKS" into 4051 memory.

COMMAND DESCRIPTIONS

PRINT

record number	<p>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.</p> <p>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.</p>
USING	See the 4051 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

NOTE

If two strings are printed in a single statement without CRs (carriage returns) the system will not be able to distinguish between them and will input them both as a single string.

1. 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.
2. Printing data to a record requires that all previous records be filled. See the file initializing program in SAMPLE PROGRAMS (Appendix F).
3. 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.
4. For examples of how to retrieve PRINTed information, see INPUT command description.
5. For programming examples, see USING PRINT AND INPUT IN A PROGRAM (Appendix G).

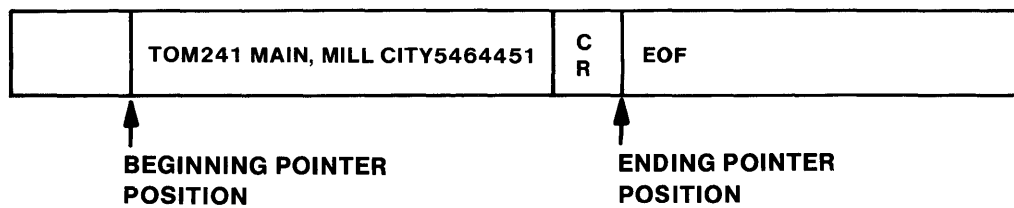
Prerequisites

Disc must be mounted but device must not be reserved. File must be open.

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 with 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"
```

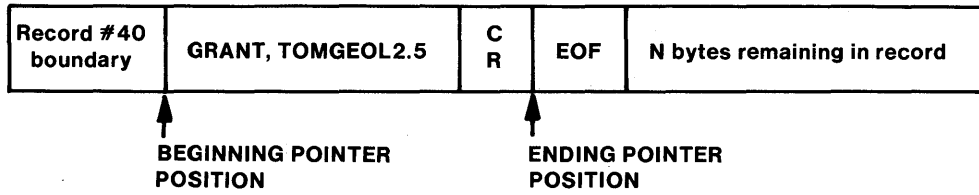
Example 3: PRINTING TO A RANDOM FILE

```
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.

COMMAND DESCRIPTIONS

PRINT



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.



Printing to a random 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. This does not apply to random binary files.

READ

Purpose

The READ command reads binary data stored in the designated file with a WRITE command. It operates like the 4051 BASIC READ command with the following exceptions:

1. 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.
2. 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.
3. The primary address character is "#".

SYNTAX FORM

$$\text{REA \# constant [,numeric expression] : } \left\{ \begin{array}{l} \text{array variable} \\ \text{string variable} \\ \text{numeric variable} \end{array} \right\} \left[, \left\{ \begin{array}{l} \text{array variable} \\ \text{string variable} \\ \text{numeric variable} \end{array} \right\} \right] \dots$$

DESCRIPTIVE FORM

$$\text{READ \# lfn [,record number] : target variables } \left[\begin{array}{l} \text{target variables} \\ \text{for incoming data} \end{array} \right] \dots$$

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.

COMMAND DESCRIPTIONS

READ

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 4051 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 H).

Prerequisites

Disc must be mounted but device must not be reserved. File must be open.

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.

SAVE

SAVE

Purpose

The SAVE command transfers a copy of the current BASIC 4051 program to a file on the disc either in 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. [,"ASCII"][; 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 match the F.I. in the CREATE command. If no file has been created, this command will create its own. |
| ASCII | If "ASCII" 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. |
| 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

1. The stored program may be retrieved with an APPEND or OLD command.
2. If speed is important, programs should be saved in binary (default) form.
3. A SAVE command may not be executed to a random file.
4. A file does not need to be opened to execute a SAVE command.
5. If the file exists prior to this command, the file type (ASCII or host 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.

Example 2:

```
350 SAVE A$,"ASCII"
```

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 form.

record number	<p>An entry is necessary here if you are writing to a random file.</p> <p>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.</p> <p>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.</p>
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.
data item	Same as previous field.

General Information

1. 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 that point.
2. For programming examples see USING WRITE AND READ IN A PROGRAM (Appendix H).

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 functions¹ 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.

WRITE

Prerequisites

Disc must be mounted but device must not be reserved. File must be open.

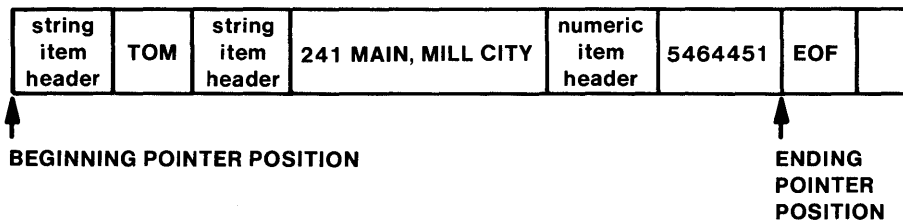
Examples

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, as shown below.



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
```

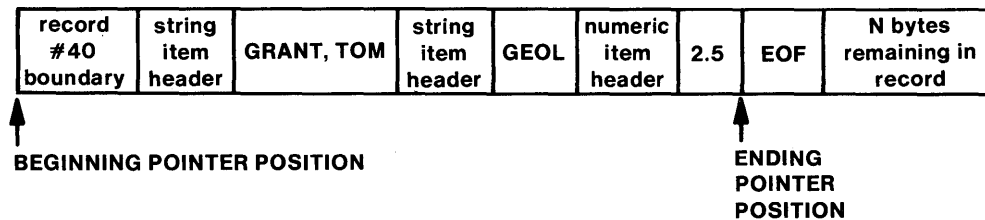
This example functions exactly as Example 2 if:

- 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 automatically "padded."

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

Section 6

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)

SPECIFICATIONS

4907 ENVIRONMENTAL SPECIFICATIONS

1. Operating Environment

Ambient Temperature	50 degrees F to 100 degrees F (10 degrees C to 38 degrees C)
Relative Humidity	20% to 80%
Maximum Wet Bulb	78 degrees F (25 degrees C)
Maximum Altitude Above Sea Level	10,000 ft (3048 m)

2. Storage Environment

Ambient Temperature	50 degrees F to 125 degrees F (10 degrees C to 52 degrees C)
Relative Humidity	8% to 80%
Maximum Altitude Above Sea Level	50,000 ft (15,240 m)
Shipping Temperature	40 degrees F to 125 degrees F (-40 degrees C to 52 degrees 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 2-1 shows the allowable line voltages:

Table 6-1

LINE VOLTAGES

Line Voltage	Tolerance	Frequency	Fuse
100 Vac	90-110	50 or 60 Hz ±.5 Hz	2 A Med Blow
120 Vac	108-132		2 A Med Blow
220 Vac	198-142		1 A Med Blow
240 Vac	216-264		1 A Med Blow

2. Maximum power dissipation at 120 Vac, 60Hz, 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

SPECIFICATIONS

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 degrees F to 140 degrees F (5 degrees C to 60 degrees C)
Humidity	8% to 80%
3. Media Lifetime	
Passes per track Insertions	3.5 x 10E5 > 30,000

ROM PACK

1. Dimensions

Length	466 in
Width	2.62 in
Depth	0.88 in

2. Weight 8 oz

3. Environmental Requirements

Same as 4907 ENVIRONMENTAL SPECIFICATIONS.

4. Power Requirements (from 4051)

+5 Vdc 300 mA

APPENDICES

CONTENTS

Appendix A	DEVICE AND FILE STATUS MESSAGES
Appendix B	ERROR MESSAGES AND RECOVERY PROCEDURES
Appendix C	FILE POINTER OPERATION
Appendix D	GLOSSARY
Appendix E	ROUTINE FLEXIBLE DISC DRIVE MAINTENANCE
Appendix F	SAMPLE PROGRAMS
Appendix G	USING PRINT AND INPUT IN A PROGRAM
Appendix H	USING WRITE AND READ IN A PROGRAM

Appendix A

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-MOM-YY HH:MM ALT
          NNNNNNN  USED      DD-MOM-YY HH:MM USED
          N OPEN     NNNNNNN REC LEN   DD-MOM-YY HH:MM CREATED
FILE IDENTIFIER
```

Field Descriptions

```
DD-MOM-YY HH:MM ALT
DD-MOM-YY HH:MM USED
DD-MOM-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"
```

APPENDIX A

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" se- quential)	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

Shows what the file attributes are, 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 CREATED
```

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 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  
XXXXXXXXXXXXX DEV ID XXXXXXXXXXXX VOL ID XXXXXXXXXXXXXXXXXXXXXXXXXXXX OWNER  
NNNNNNNNNN FREE NNNNNNNNNN SIZE NNNNNNNNNN 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.
XXXXXXXXXXXXX DEV ID	Shows the nomenclature of the device.
XXXXXXXXXXXXX VOL ID	Indicates the name of the disc as specified in the CALL "FORMAT" command.
XXXXXXXXXXXXXXX OWNER	Indicates the owner's or user's name as specified in the CALL "FORMAT" command.
NNNNNNNNNN FREE	Indicates the number of usable non-file space remaining on disc.
NNNNNNNNNN SIZE	Shows the number of storage bytes provided by this type of disc.
NNNNNNNNNN LOST	Indicates the number of bytes of unusable or damaged space.
NNNNN BLK SIZE	Indicates the size of each principal block.
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      INCTAX      VOL      ID
SMITH & CO. OWNER  250000 FREE    630000 SIZE
2560 LOST 256 BLK SIZE 07-MAR-77 10:30 FORMATTED
00003 OPEN

```

This message would be generated for device 1, which is identified as a 4907. The disc is identified as "INCTAX" and the owner as "SMITH & CO." There are 250,000 bytes of usable non-file space. The 630,000-size field indicates this is a flexible disc with 2,650 bytes of unusable or damaged space. The block size is 256 bytes.

The disc was last formatted on March 7, 1977 at 10:30 AM. Three files on the disc are open, and the device is not reserved or write-protected.

APPENDIX B

Appendix B

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, re-starting program may be necessary.
ERROR 2 ILLEGAL COMMAND	4907 or 4051 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 Service Manual.
ERROR 3 COMMAND FORMAT	4907 or 4051 error.	See 4907 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.

APPENDIX B

Error Message	Cause	Correction
<p>ERROR 5 ILLEGAL MASTER PASSWORD</p>	<p>An attempt has been made to execute a command containing an illegal entry in the master password field.</p>	<p>Master password field can contain no more than 10 characters. The first character must be alpha and the rest alphanumeric. No spaces may be entered between characters.</p>
<p>ERROR 7 ILLEGAL FILE IDENTIFIER</p>	<p>An F.I. has incorrect construction or illegal characters.</p>	<p>See FILE IDENTIFIER CONSTRUCTION in Section 4.</p>
<p>ERROR 8 ILLEGAL VOLUME IDENTIFIER</p>	<p>The volume I.D. field in a formatting command has illegal characters, incorrectly located spaces, or is blank.</p>	<p>The volume I.D. field can contain no more than 10 characters. The first character must be alpha. No spaces may be entered between characters. The field cannot be blank.</p>
<p>ERROR 10 DEVICE NOT FOUND</p>	<p>The system cannot find the device with the address specified in the command.</p>	<p>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.</p>
<p>ERROR 11 DEVICE WRITE PROTECTED</p>	<p>An attempt was made to write to a disc that is write-protected.</p>	<p>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.</p>

Error Message	Cause	Correction
<p>ERROR 12 DEVICE RESERVED</p>	<p>The device was reserved and a command requiring a free device was attempted.</p>	<p>Release the device with a CALL "DREL" command and continue.</p>
<p>ERROR 13 DEVICE NOT RESERVED</p>	<p>The device was not reserved and a command requiring a reserved device was attempted.</p>	<p>Execute a CALL "DRES" command and continue.</p>
<p>ERROR 14 DEVICE HAS FILES OPEN</p>	<p>A CALL "DRES" command was issued to a device with open files.</p>	<p>Close all files on that device and continue.</p>
<p>ERROR 15 DEVICE I/O ERROR</p>	<p>A hard error has occurred on a device.</p>	<p>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 Service Manual for complete details.</p>
<p>ERROR 16 DEVICE NOT READY*</p>	<p>This message appears if:</p> <ol style="list-style-type: none"> 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. 	<p>Make sure address is correct, that the disc is properly loaded, and that the device is up to speed.</p>
<p>ERROR 17 DEVICE NOT MOUNTED*</p>	<p>An attempt was made to use an unmounted disc.</p>	<p>Execute a CALL "MOUNT" command and continue</p>

APPENDIX B

Error Message	Cause	Correction
<p>ERROR 18 NO SPACE*</p>	<p>There is not enough space left on the disc for further storage.</p>	<p>Delete any unneeded files (KILL), gather the remaining free space in a single area and with the CALL "COMPRS" command II/O command may then be reissued.</p>
<p>ERROR 20 CONTROL UNIT ERROR TABLE SPACE EXHAUSTED</p>	<p>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.</p>	<p>Close two files and reissue the command.</p>
<p>ERROR 22 CONTROL UNIT PROCESSOR ERROR RAM FAILED</p>	<p>A hardware failure in the controller prevents further system use.</p>	<p>See 4907 Service Manual.</p>
<p>ERROR 23 CLOCK NOT READY</p>	<p>The system is not operating.</p>	<p>Execute a CALL "SET-TIM" command</p>
<p>ERROR 32 DIRECTORY CHAIN DAMAGE*</p>	<p>This may occur when attempting a CALL "MOUNT" command.</p>	<p>Execute CALL "DUP". Then reattempt CALL "MOUNT" on new disc. Information may be unrecoverable on a disc generating this message.</p>

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: <ol style="list-style-type: none"> 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: <ol style="list-style-type: none"> 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 earlier OPEN command.	Execute an OPEN command with the correct lfn.

APPENDIX B

Error Message	Cause	Correction
<p>ERROR 41 ACTIVE LOGICAL FILE NUMBER*</p>	<p>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.</p>	<p>Execute another OPEN command with an unused lfn, or close old file.</p>
<p>ERROR 42 FILE NOT FOUND*</p>	<p>The device involved does not contain a file with the specified name.</p>	<p>Be sure the device address and the F.I. are correct and reissue the command.</p>
<p>ERROR 43 DUPLICATE FILE IDENTIFIER*</p>	<p>An attempt was made to create a new file with an F.I. belonging to an existing file.</p>	<p>Use a different F.I. in the CREATE command.</p>
<p>ERROR 44 LIBRARY NAME CONFLICT</p>	<p>A file cannot be created at any level where there is already a library of the same name.</p>	<p>Change the name of the file and reissue the CREATE command.</p>
<p>ERROR 45 FILE LOCKED*</p>	<p>An attempt was made to open a file without using the passwords specified in the CREATE command.</p>	<p>Reissue the OPEN command using the correct passwords.</p>
<p>ERROR 50 FILE IS WRITE PROTECTED*</p>	<p>An attempt was made to WRITE to a file opened for read access only.</p>	<p>If you do wish to write to the file, close it and then reopen it for "full" access.</p>

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 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 Service Manual.

APPENDIX B

Error Message	Cause	Correction
<p>ERROR 57 ILLEGAL ITEM HEADER*</p>	<p>The TYPE function has encountered an illegal item header for a standard item file.</p>	<p>See TYP command description in Section 5.</p>
<p>ERROR 60 FORMAT FAILED*</p>	<p>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.</p> <p>If the CALL "FFRMT" command was issued, it may have failed because the old volume label was illegible due to damage.</p>	<p>If message appears after attempting a complete, accurate "full" format, choose another disc.</p>
<p>ERROR 61 MARK BAD BLOCK GROUP FAILED*</p>	<p>The CALL "MRKBBG" command failed because the bad block table in the volume label is too full or hardware errors are involved.</p>	<p>Copy files to a new disc or refer to 4907 Service Manual.</p>
<p>ERROR 62 ILLEGAL ATTRIBUTE CHANGE*</p>	<p>The ASSIGN command contains attribute entries which conflict with the existing file type.</p>	<p>Reissue the ASSIGN command with correct attribute entries.</p>

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 password 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.

APPENDIX B

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 "RE- NAME" command with the correct passwords.

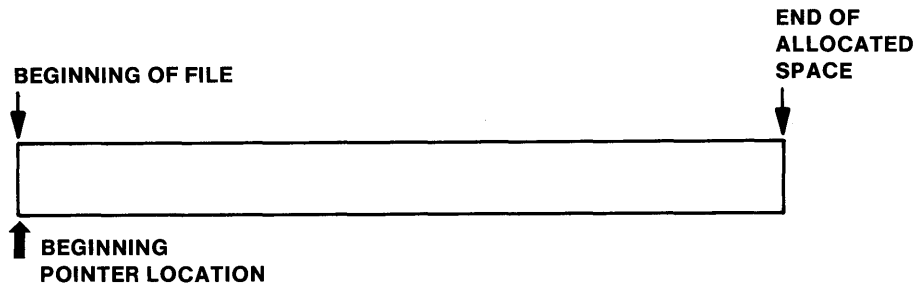
APPENDIX C

Appendix C

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.

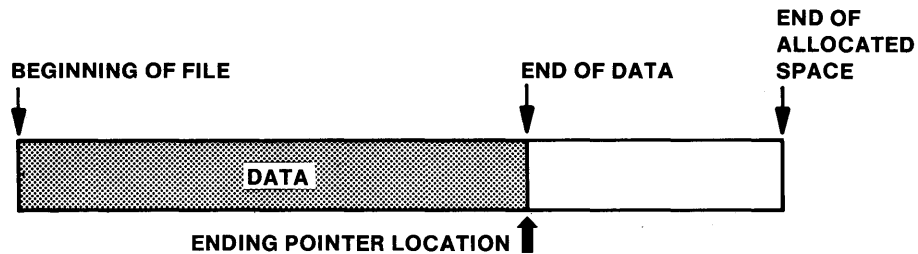
Each file is equipped with a "pointer." Although this pointer is invisible to the user, it indicates to the 4051 where an I/O operation such as a READ or WRITE is to begin. When a file is created, the pointer is automatically positioned at the beginning of the file. Where the pointer is positioned after 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.



2380-38

Figure C-1. Graphic Representation of Files.

Once data has been entered, the file is closed. The pointer remains where it was when data entry ended (Figure C-2).



2380-39

Figure C-2. Pointer location at the end of data entry after an OPEN "F".

APPENDIX C

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.

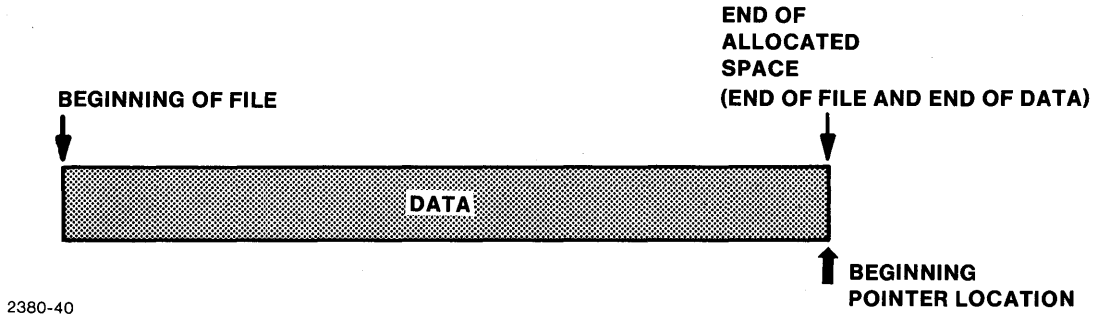


Figure C-3. Pointer location at the end of data entry after an OPEN "U".

If an "R" is entered instead of a "U," the pointer is located at the beginning of the file (Figure C-4).

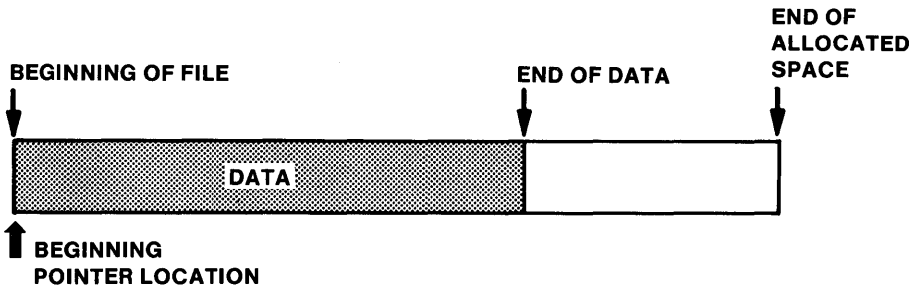


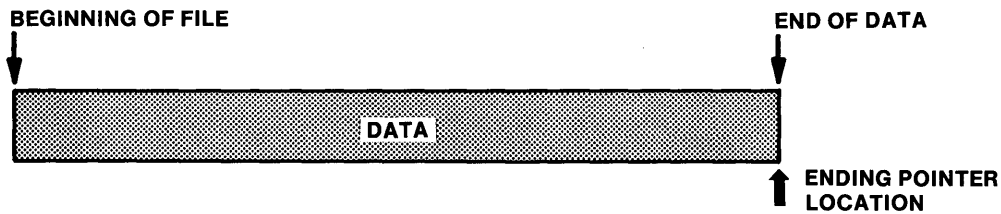
Figure C-4. Pointer location prior to I/O command after an OPEN "R".

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.

¹When an EOF is encountered in a program, the program will abort unless an ON EOF command was executed previously.

Rewinding

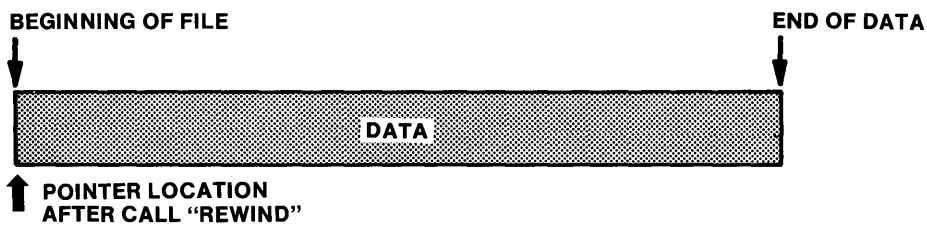
Normally, the pointer cannot be moved backward to the beginning of a sequential file unless a CLOSE and a subsequent OPEN are carried out. There are, however, situations when convenient pointer movement is desirable, for example, when you wish to READ or reenter data just entered. A CALL "REWIND" command 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.



2380-42

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).



2380-43

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 tables below 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

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 the TEKTRONIX 4051 Graphic System.
BIT	A binary digit; a 1 or a 0.
BYTE	A group of eight binary bits.

APPENDIX D

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.
COMMAND	A line of instructions containing a keyword(s), address(es) and/or an argument. Usually issued from 4051 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 of 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 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 characters or spaces 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.

DIMENSIONING	Enlarging or reducing of string variable space to accommodate more or less than the 72 character default length.
DEVICE, RESERVED	A device reserved for formatting and other device management activities.
DISC	The recording medium used within a drive. Also may be referred to as the "volume", or when residing in the drive, the "device."
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.

APPENDIX D

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, 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 standard 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 4907 host (4051) controls all system activity with the exception of system front panel switches.
HOST BINARY	The 4051 saves programs in either ASCII or binary form. The binary form is host binary which means it can be entered only by another host (4051). 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 program listings.
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 4051 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 4051 used to contain programs.
NUMERIC EXPRESSION	See 4051 Graphic System Reference Manual.
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 USING PASSWORDS IN FILE IDENTIFIERS in Section 4.
POINTER	See FILE POINTER OPERATION (Appendix C).
RANDOM FILE	A type of file with two or more records of identical length.
RANDOM ACCESS	The method of storing or locating data by specifying a record number and directly accessing that record only. Opposite of sequential access.
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.

APPENDIX D

SCRATCHLIB	The system name of the default current library.
SEQUENTIAL FILE	A type of file with no divisions or "records."
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. Opposite of random access.
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 FILE IDENTIFIERS 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 4051 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 (4051).
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 4051 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).

APPENDIX E

Appendix E

ROUTINE FLEXIBLE DISC DRIVE 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. Check for dust and dirt.	12 months	Tighten screws, connectors, and switches. Clean off dust and dirt.
6	Read/write head	Aborted I/O commands or distorted results.	12 months	Align head.

APPENDIX E

Procedure No. 1

- a. Remove cover from drive (Figure E-1).



Figure E-1. Removing Main 4907 Cabinet Cover.

- b. Remove oxide from head with swab and alcohol (Figure E-2).

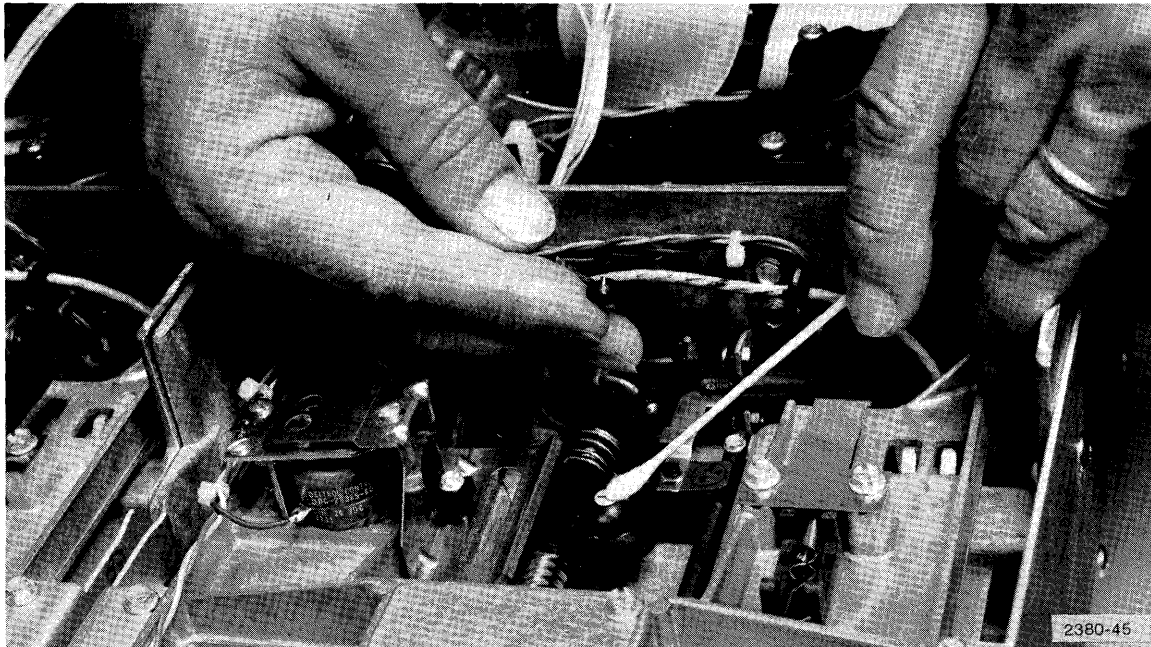


Figure E-2. Cleaning Read/Write Head.

- c. Replace drive cover.

APPENDIX E

Procedure No. 2

- a. Remove cover from drive (Figure E-3).



Figure E-3. Removing Main 4907 Cabinet Cover.

- b. Remove read/write head button (Figure E-4).

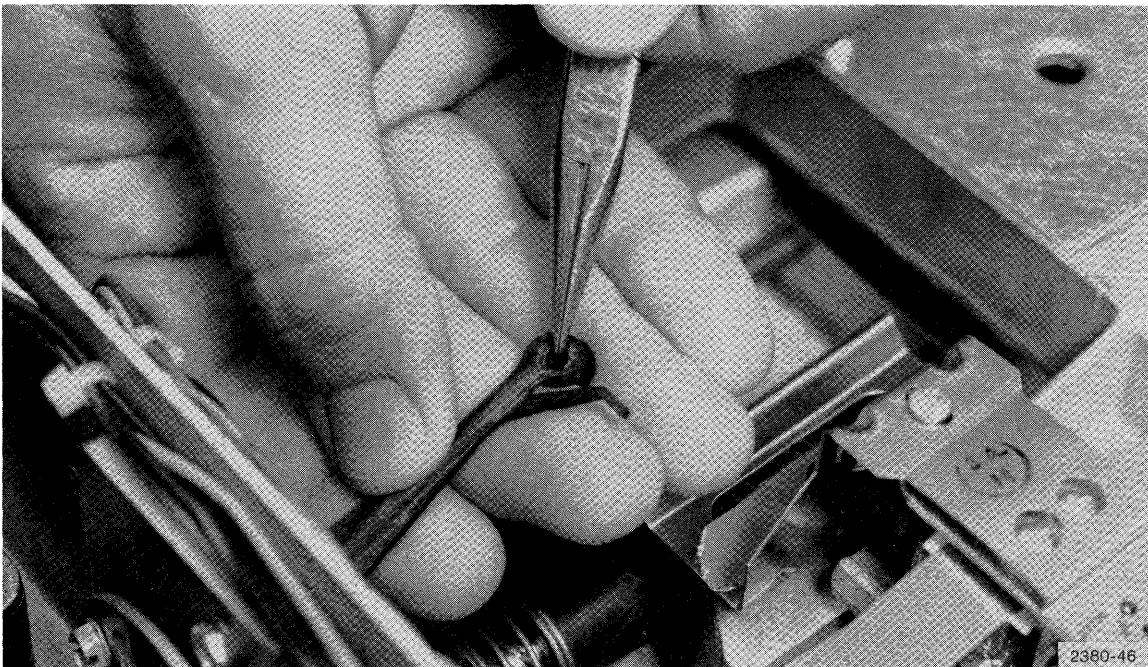


Figure E-4. Removing Read/Write Head Button.

- c. Install new read/write head button (Figure E-5).

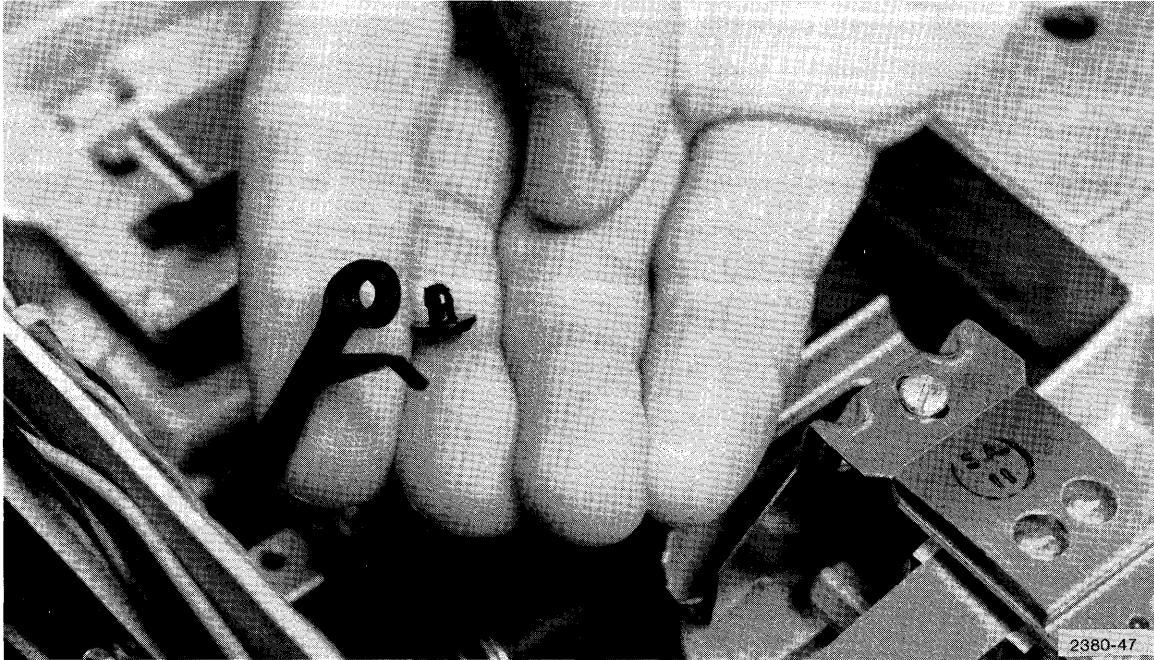


Figure E-5. Installing New Read/Write Head Button.

- d. Replace drive cover.

APPENDIX F

Appendix F

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:

- a. Initialize the 4051.
- b. Designate the current device.
- c. Mount the device.
- d. Designate the current library (CALL "USERLIB").
- e. Set the system clock.
- f. Specify the F.I.
- g. Kill (delete) the file if desired.
- h. Create a random file and establish the number of records and record length. See CREATE command description for details on determining record length.
- i. Create a sequential file if desired.
- j. Assign file attributes.
- k. Open the file.
- l. Print to the file if desired (ASCII).
- m. Write to the file if desired (binary).
- n. Enter blank strings in all records of random files. See step 9 of GENERAL SEQUENCE FLOW CHART in Section 2.

```
100 PRINT "RANDOM FILE INITIALIZATION PROGRAM"
110 INIT
120 PRINT "ENTER DEVICE ADDRESS:";
130 INPUT A
140 CALL "UNIT",A
150 CALL "MOUNT",A,""
160 PRINT "ENTER LIBRARY OR LIBRARIES (DOG/CAT):";
170 INPUT A$
180 IF A$="" THEN 200
190 CALL "USERLIB",A$
200 CALL "TIME",A$
210 IF A$<>"" THEN 250
220 PRINT "ENTER DATE AND TIME:";
230 INPUT A$
240 CALL "SETTIM",A$
250 PRINT "ENTER NAME OF FILE:";
260 INPUT A$
```

APPENDIX F

```

270 CALL "FILE",A,A$,B$
280 IF B$="" THEN 330
290 PRINT "IF FILE EXISTS DO YOU WANT IT DELETED? (Y OR N):";
300 INPUT B$
310 IF B$<>"Y" THEN 250
320 KILL A$
330 PRINT "ENTER RECORD LENGTH:";
340 INPUT A
350 PRINT "ENTER NUMBER OF RECORDS:";
360 INPUT B
370 PRINT "IS THIS AN ASCII FILE? (Y OR N):";
380 C$="A"
390 INPUT B$
400 IF B$="Y" THEN 420
410 C$=""
420 PRINT "OTHER ATTRIBUTES THAN ";C$;"RCN? ENTER CHARACTERS, IF ANY:";
430 INPUT B$
440 B$=B$&C$
450 CREATE A$,B$;B,A
460 IF A=0 THEN 620
470 OPEN A$;1,"F",B$
480 DIM D$(A-1)
490 D$=""
500 FOR I=1 TO A-1-4*(C$<>"A")
510 D$=D$&" "
520 NEXT I
530 IF C$="A" THEN 580
540 FOR I=1 TO B
550 WRITE #1,I:D$
560 NEXT I
570 GO TO 610
580 FOR I=1 TO B
590 PRINT #1,I:D$
600 NEXT I
610 CLOSE 1
620 END

```

2. ASCII DATA TRANSFER FROM TAPE TO DISC FILE

This program prints ASCII data from an internal 4051 magnetic tape to a specified sequential disc file.

```

100 INIT
110 PRINT "IGGGTAPE TO DISC PROGRAM FOR DEVICE 0"
120 UNIT 0
130 CALL "MOUNT",0,""
140 PRINT "JJENTER LIBRARY OR LIBRARIES (DOG/CAT):";
150 INPUT A$
160 IF A$="" THEN 180
170 CALL "USERLIB",A$
180 CALL "TIME",A$
190 IF A$("<")="" THEN 230
200 PRINT "DATE AND TIME:";
210 INPUT A$
220 CALL "SETTIM",A$
230 PRINT "ENTER SOURCE FILE NUMBER FROM TAPE:";
240 INPUT N
250 IF N=0 THEN 420
260 FIND N
270 PRINT "ENTER NAME OF DISC FILE ONLY:";
280 INPUT A$
290 CALL "FILE",0,A$,B$
300 IF B$="" THEN 350
310 PRINT "SHOULD EXISTING FILE DATA BE ERASED? (Y OR N):";
320 INPUT B$
330 IF B$="Y" THEN 370
340 GO TO 370
350 CREATE A$,"A";256,0
360 REM FILE SIZE WILL BE INCREASED AS NECESSARY
370 OPEN A$;1,"F",B$
380 ON EOF (0) THEN 420
390 T=TYP(0)
400 GO TO T OF 430,450,480,510
410 PRINT "EMPTY FILE"
420 GO TO 540
430 PRINT "END OF FILE"
440 GO TO 540
450 INPUT @33:A$
460 PRINT #1:A$
470 GO TO 390
480 READ @33:A
490 PRINT #1:A4
500 GO TO 390
510 READ @33:A$
520 PRINT #1:A$
530 GO TO 390
540 END

```

APPENDIX G

Appendix G

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.

APPENDIX H

Appendix H

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.

INDEX

Accessories, standard and optional	1-6, 1-7
Address	4-1
Address strapping	2-1
Allocated space	5-55, 5-84
Argument	4-2
ASCII	5-97, D-1
BASIC	ix, xiii, 1-1, 4-1, D-1
Binary	5-106
Blocks	5-40, A-4
Braces	5-3
Brackets	5-2
Bytes	5-65, 5-84, A-1, A-4, D-1
Chains	5-33
Changing names	5-43, 5-53
Clock	2-8, 2-18, 5-9, 5-11, A-1
Colons	4-5, 4-26
Command delimiters	4-5
Command descriptions	
APPEND	5-89
ASSIGN	5-43
CLOSE	5-45
COMPRESS (CALL "COMPRS")	5-19
CONTROLLER/UNIT STATUS (CALL "CUSTAT")	5-21
COPY ... TO	5-47
CREATE	5-54
DELETE ALL	5-5
DEVICE RELEASE (CALL "DREL")	5-25
DEVICE RESERVE (CALL "DRES")	5-27
DEVICE STATUS MESSAGE (CALL "DSTAT")	5-29
DIRECTORY	5-59
DISMOUNT (CALL "DISMOUNT")	5-23
DUPLICATE (CALL "DUP")	5-62
END	5-64
FAST FORMAT (CALL "FFRMT")	5-31
FILE MANAGER VALUES (CALL "FMVALS")	5-6
FILE STATUS (CALL "FILE")	5-65
FORMAT (CALL "FORMAT")	5-32
HARD ERROR STATUS (CALL "HERRS")	5-36

INDEX

INITIALIZE.....	5-8
INPUT.....	5-92
KILL.....	5-67
MARK BAD BLOCK (CALL "MRKBBG").....	5-40
MOUNT (CALL "MOUNT").....	5-38
NEXT (CALL "NEXT").....	5-69
OLD.....	5-95
ON END-OF-FILE.....	5-71
OPEN.....	5-73
PRINT.....	5-97
READ.....	5-101
RENAME (CALL "RENAME").....	5-78
REWIND (CALL "REWIND").....	5-81
SAVE.....	5-104
SECRET.....	5-83
SET TIME (CALL "SETTIM").....	5-9
SPACE (CALL "SPACE").....	5-84
TIME (CALL "TIME").....	5-11
TYPE (Function).....	5-86
UNIT.....	5-12, 5-14
USER LIBRARY (Current library).....	5-16
WRITE.....	5-106
Command punctuation.....	4-5
Command examples.....	4-6
Command summaries.....	2-11
Commas.....	4-5
Configuration, 4907.....	1-1
Configuration, 4907 Option 30.....	1-2
Configuration, 4907 Option 31.....	1-2
Constant.....	4-3, D-2
Contiguous files.....	5-55
Controller.....	1-1
Controller/device commands.....	5-19
Controls and indicators.....	1-3
Current device.....	5-12
Current library.....	5-16
Delimiters, command.....	4-5
Delimiters, file identifier.....	4-26
Descriptive form.....	5-3
Device (also see Drive).....	5-12, D-2
Device status message.....	A-4
Dimensioning.....	4-4
Disc, see specifications, flexible disc	
Drive (also see Device).....	1-1, 6-3

Error messages	B-1
Extending files	2-19, 5-84
Extensions	4-25
Features	ix, 2-18
Field descriptions, command	4-1
Field descriptions, F.I.	4-16
Fields	4-1, 4-16
File description	2-17, 3-1
File extending	2-19, 5-84
File identifier	1-3, 4-12
File length	5-54
File management commands	5-43
File pointer operation	C-1
File status message	A-1
Files	2-17, 3-1
Files, contiguous	5-54
Files, sequential	2-17
Flexible disc loading	2-4
Flexible disc specifications	6-4
Flexible disc storage and handling	2-6
Flow chart, general sequence	2-8
GPIB cable	2-2
Group OPEN	2-20, 5-73
Host	x
Instrumentation	xii
I/O commands, file	5-89
I/O program	2-15
Keyword	4-1
Length, file	5-54
Levels	3-1, 3-2, 4-16
lfn (see Logical File Number)	
Libraries	3-2, 3-3, 4-12
Line numbers	5-3
Loading, flexible disc	2-4
Logical File Number	4-2, 5-73
Maintenance, routine flexible disc drive	E-1
Master password	5-31, 5-32
Message, device status	A-4
Message, file status	A-1
Multiple file selection	4-27
Multiple file name changing	5-53
Multiple file transfers	5-49
Numeric expressions	4-3
Numeric variable	4-4

INDEX

Option 10	5-59
Passwords	4-22
Peripherals.....	xiii
Pointer (see File pointer operation)	
Power-up	2-3
Preparation	2-1
Prerequisites.....	5-3
Printer	xiii, 5-59
Programming.....	xii, 5-92, 5-97
	G-1, H-1
Programs, sample.....	2-15, F-1
Punctuation, command.....	4-5
Questions and answers	xi
Quotation marks	4-5, 4-26
Random access (also see I/O commands, file).....	2-18, 2-20
Random files	2-18
Records.....	2-18, 5-54
Recovery procedures	B-1
Releasing a device.....	5-25
Reserved device.....	5-27
ROM Expander (4051E01)	2-3
ROM pack.....	1-1, 2-2, 6-5
Sample I/O program	2-15
Sample programs.....	2-15, F-1
Scattered files.....	5-54
Scratch library (SCRATCHLIB)	4-17, 4-21
Selection, multiple file	4-27, 4-38
Semicolons	4-5
Sequential access (also see I/O commands, file)	2-17
Sequential files.....	2-17
Simultaneous file use	2-20
Spaces	4-5
Special characters.....	2-19, 4-27
Specifications, 4907 performance	6-1
Specifications, 4907 physical	6-1
Specifications, electrical	6-3
Specifications, environmental	6-2
Specifications, flexible disc	6-4
Status messages	2-19, A-1
Storage structure.....	3-1
Strapping, address.....	2-1
String constants.....	4-3
String variables.....	4-4
Surface analysis.....	5-31, 5-32

Syntax	4-3
Syntax form	5-2
System clock	2-8, 2-18, 5-9 5-11, A-1
System commands	5-5
System library (SYSLIB)	4-17
Target numeric variable	4-5
Target string variable	4-4
Trailing dots	4-6
Transfers, multiple files	5-49
User library	4-17, 4-20
Variables	4-4, 4-5
Volume	D-7
Volume label	5-31, 5-32, 5-62
Write-protect hole	2-4
Write-protect switches	1-3, 1-4, 1-5