List of Software Versions
FDOS ~~1.5~~ 1.6 CHANGE #1 5/82

1720A
Instrument Controller

Floppy Disk Operating System User Manual

P/N 633123
December 1981

FLUKE®

CHANGE/ERRATA INFORMATION

ISSUE NO: 1 5/82

This change/errata contains information necessary to ensure the accuracy of the following manual.

MANUAL

Title:	1720A Floppy Disk Operating System User Manual
Print Date:	December 1981
Rev. and Date:	---

C/E PAGE EFFECTIVITY

Page No.	Print Date
1	5/82

Table of Contents

List of Illustrations

List of Tables

Section 1
Introduction

1-1. INTRODUCTION

1-2. The Floppy Disk Operating System (FDOS) is a machine-language system program supplied on the System Disk with the file name FDOS.SYS. It responds to interrupts from input and output devices, such as the keyboard, the touch-sensitive display, and the floppy disk. In addition, FDOS performs file management on the floppy and electronic disks for all other programs. Whenever the 1720A is operating under normal conditions, FDOS is present in Main Memory. FDOS also includes a processor that allows command files of stored user defined keyboard inputs to be processed automatically when called. Since these command files consist of a series of inputs to the console monitor, this concept is explained in the Console Monitor User Manual.

1-3. The 1720A is designed to be a single-user, single-processor computer so FDOS does not have many of the complexities inherent in operating systems used by large mainframe processors. Even so, FDOS is a sophisticated operating system (O/S), well-suited for its purpose as the 1720A Instrument Controller executive.

1-4. Section 2 of this manual provides user information and Section 3 provides programmer information. Appendix A and Appendix B support the programmer information.

1-5. CONVENTIONS USED IN THIS MANUAL

1-6. Introduction

1-7. The following paragraphs describe conventions used in this manual. Individual sections of this manual may define additional conventions.

1-8. Command Definitions

1-9. Each 1720A command is defined in a standard format that allows a maximum of 80 characters in any command line.

- The command is named on a title line.
- A syntax diagram or syntax statement follows the title line.

- The command definition follows the syntax diagram or syntax statement.

  1. The body of the command definition is a short paragraph that describes the basic function of that command.

  2. Amplifying information is separated by item and organized in a top down outline format similar to this description.

  3. A cross-reference to any associated information or documents ends the command definition.

1-10. Syntax Diagrams

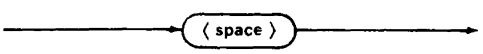
1-11. Syntax diagrams define correct spelling, punctuation, and sequence of words, symbols, and expressions for system and utility commands. The following guidelines define proper use of these diagrams:

- Any path through a diagram starting from the left that does not run contrary to an arrowhead forms a legitimate command construct. The text accompanying the diagram explains legal usage.

- Boldface words in a circular enclosure are to be entered exactly as shown. Example:

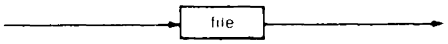
- Key entries with names, such as ESC or RETURN, are shown in a box with rounded corners. Example:

- A required space character entry is always shown as:

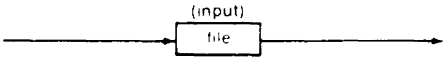


- Control character entries are shown in circular enclosures within angle brackets. The representation CTRL/ means to hold the CTRL key depressed while typing the character that follows. Example:

0 Lower case words enclosed in a box represent other information to be supplied. Example:



0 Words outside the path of the diagram, usually in parentheses, provide supplementary information. These words are normally not part of the definition of the statement. Example:



1-12. Notation Conventions

1-13. There are several notation conventions used in the text and in the examples.

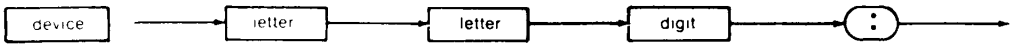
- 0 <XX> is understood to mean press key XX.
  1. <cr> is understood to mean a carriage return (the RETURN key).
  2. <lf> is understood to mean line feed.
- 0 [xxx] is understood to mean that xxx is an optional input.
- 0 {xxx} is understood to mean that xxx is a required input.
- 0 ^ is understood to mean that the characters that follow are a superscript. For example, 6^2 means six raised to the second power or 36.

Section 2
User Information

2-1. INTRODUCTION

2-2. The information in this section provides the user with the information necessary to use FDOS.

2-3. DEVICE NAMES



2-4. FDOS uses symbolic names to identify the source and destination of any information transfer, except for IEEE-488 bus instruments. Table FD-1 summarizes these device names.

- Two letters followed by a numeric digit and a colon are required.
- When a file device is specified (MF0: or ED0:), the device name is used as a prefix to a file name.

  Example: MF0:TEST.Q44.

- When no device is specified, the default for file operations is the system device (SY0:). Other defaults are defined individually.
- SY0: is always redundant and does not need to be specified.
- Usage of device names is identical for any utility program or programming language.

2-5. THE SYSTEM DEVICE

2-6. The System Device is a useful default provided in FDOS for file operations.

- When a file name is specified without a device, the System Device is assumed.
- Normally the System Device is the floppy disk (MF0:).
- The System Device can be reassigned to the electronic disk by a FUP command. This process can be automated at start-up in a command file.

- Table 2-1 lists the available devices and the entry that must be made to specify the System Device.

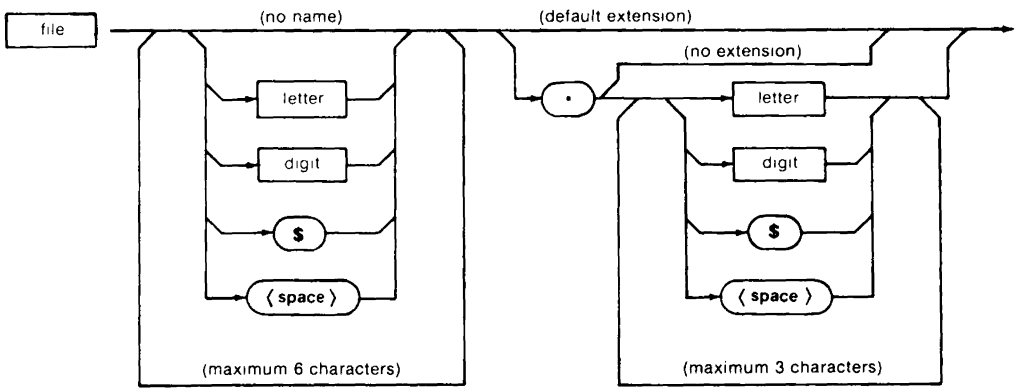
2-7. The System Device is automatically defined as the optional electronic disk at start-up time if the following conditions are met:

1. The file FDOS.SYS is not found on the floppy disk.
2. The electronic disk is installed.
3. The electronic disk contains a valid directory and the file FDOS.SYS.

Table 2-1. Device Names

ENTRY	DEVICE	ENTRY	DEVICE
SY0:	System Device	KB0:	Keyboard and Display
MF0:	Floppy Disk	KB1:	RS-232 Serial Port 1
ED0:	Optional Electronic Disk	KB2:	RS-232 Serial Port 2
MM0:	Main Memory (FUP only)		

2-8. FILE NAMES



2-9. FDOS uses file names to identify programs and data files stored on a file device (floppy or electronic disk).

- A file name is 1 to 6 letters, numeric digits, or $ or space characters, in any combination. Examples:

  TESTB2, $25795

- The optional file name extension is a period followed by 1 to 3 letters, numeric digits, or $ or space characters in any combination. Examples:

  TEST.T43, FILE9.$3

- Since a file is uniquely identified by its name, there can be only one occurrence of any file name on each file device.

- When FDOS is unable to identify a file name from the context of an operation, it searches for or assigns, a file with no name. There can be only one such no-name (all-space) file on each file device.

- A period at the end of a file name without an extension specifies a blank (3-space) extension.

- A file name without an extension specifies a default extension that is defined by the utility program or language in use.

2-10. The following file names are used by FDOS to identify special purpose files:

NAME	MEANING
FDOS.SYS	Floppy Disk Operating System
CONMON.SYS	Console Monitor
TIME.CIL	Time and Date Set Utility
SET.CIL	Set Serial Ports Utility
FUP.CIL	File Utility Program
BASIC.CIL	Fluke Enhanced BASIC Interpreter
COMMND.SYS	System Start-up Command File
FUP.HLP	File Utility Program Help Display File

2-11. The following file name extensions are used by FDOS to identify certain types of files:

EXTENSION	MEANING
.SYS	System-level programs and command file
.CIL	Utility programs (Core Image Load)
.BAS	ASCII text BASIC program
.BAL	Lexically analyzed BASIC programs
.HLP	System level data files
.CMD	Utility command files

2-12. COMMAND FILES

2-13. Introduction

2-14. A command file is a sequence of Programmer Keyboard inputs stored in a file. Command files are identified by the form of the file name. When processed, the contents of a command file are treated as if they were input from the Programmer Keyboard, except for the ? and & characters which identify special operations.

2-15. Two types of command files are available:

1. COMMND.SYS is the system command file. It operates only at start-up time.

2. (File name).CMD is a utility command file. It operates whenever its file name is given as input to the Console Monitor.

2-16. The following characteristics are common to all command files:

- All 1720A keyboard operations are accessible from a command file.
- Command file processing begins from the Console Monitor.
- Each line contains one keyboard input command.
- Command file lines are processed in sequence only.
- The & character identifies a line to be sent directly to the display. It may include display control sequences to format a message.
- If a display line includes either the & or ? characters, each must be preceded by a backslash (\).
- Only one command file can be active at any time.
- Command files can only be one block long.
- When a command file is active, a <CTRL/C> entry from the keyboard is transmitted to the running program. Once the running program has processed the <CTRL/C> entry, the command file is aborted and "Command file aborted" appears on the display.

2-17. A command file remains active until one of the following occurs:

1. Its last line has been processed.

2. Control has been passed to another utility command file.

3. An error has been encountered.

2-18. Whenever a command file is active, it is the only source of keyboard input. This means that an INPUT statement in a user program, called by a command file that has additional lines, will take its input from the next line of the active command file. If the program terminates through an END, STOP, or EXIT statement, the next command file line is taken as keyboard input.

2-19. A command file may transfer control to the first line of any utility command file, including itself, by commanding a return to the Console Monitor, followed by a line containing the name of the utility command file. The .CMD extension is not used. Subsequent command lines in the original command file will not be processed.

2-20. The System Command File

2-21. The system command file is activated only at system start-up time. After power-on, or pressing RESTART, the 1720A runs an internal self-test, loads FDOS, and searches the start-up system device for a file named COMMND.SYS. If it is found, the Console Monitor is then loaded with command file status active, and FDOS processes the command file as keyboard inputs to CONMON and any other mode selected as a result of the inputs. The system command file takes control only at start-up time. Control cannot be subsequently transferred to it. Figure 2-1 illustrates this sequence.

2-22. Utility Command Files

2-23. A utility command file is a sequence of keyboard inputs stored in a file with a .CMD name extension. The effect is that the user can create simplified commands that operate directly from the Console Monitor.

2-24. Control is transferred to a utility command file whenever the Console Monitor processes a file name which is found to have the .CMD extension. In addition, a command argument can be passed to a utility command file at the time it is called. It is generally good practice to include commands that return the system to the Console Monitor after completing the task.

2-25. Utility command files have all of the characteristics listed previously. The following points are true only for utility command files:

- The file name extension .CMD identifies a utility command file.
- Control is transfered to a utility command file from the Console Monitor by entering the name of the command file. The .CMD extension is not used.
- A utility command file may be called directly or from any active command file, including itself.
- An argument can be passed to a utility command file by including additional characters after the file name.
    1. These characters must be separated from the file name by a space and must represent a valid keyboard input for the operation that will use them.
    2. Wherever a ? character appears in the command file, the argument is substituted. The ? character is ignored when an argument is not included.
- The following paragraphs show command file examples.

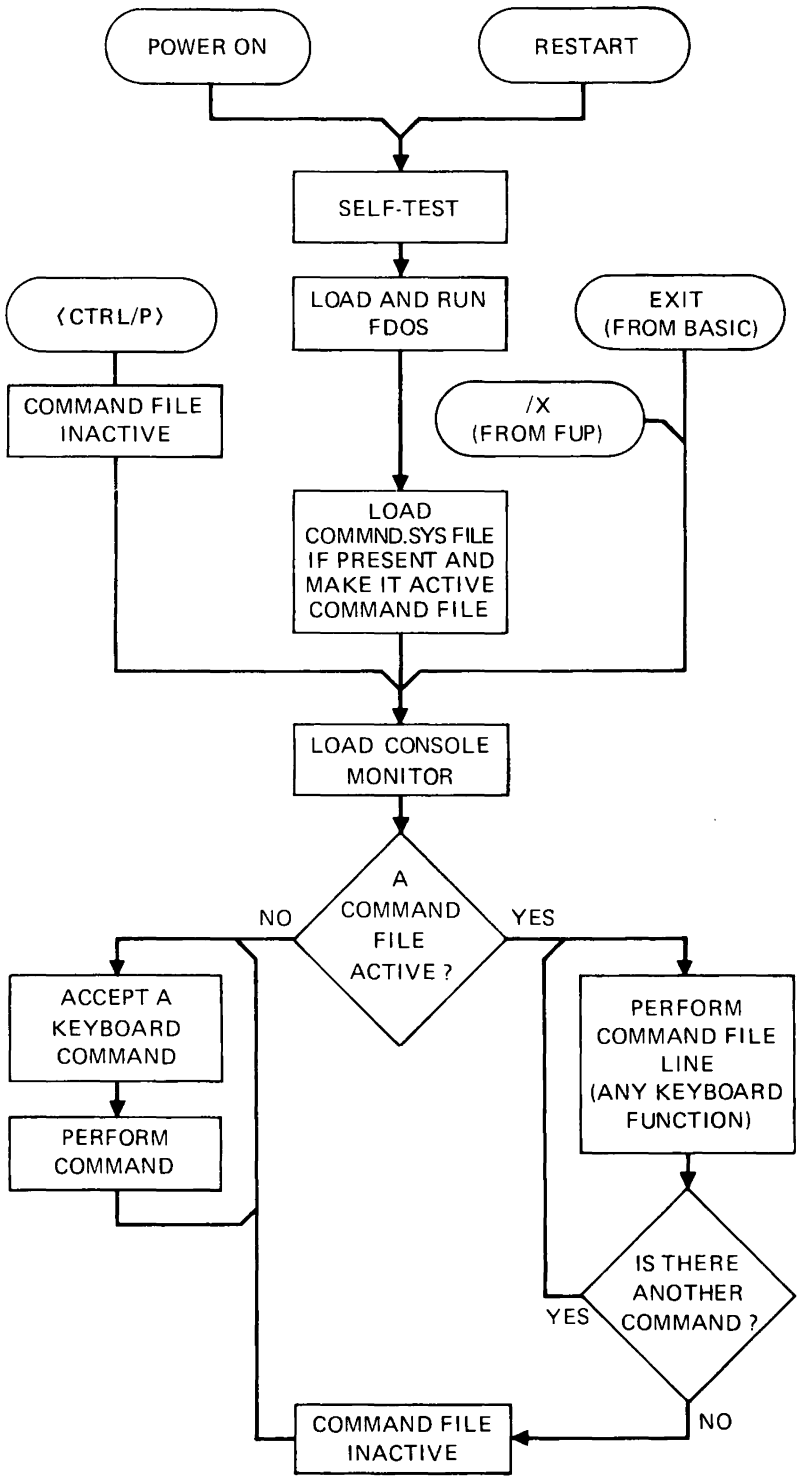


Figure 2-1. Start Sequence

2-26. Command File Examples

2-27. The following command file performs four tasks:

1. Sets some selected parameters on Serial Port 2 using the SET utility.

2. Verifies that the time clock has not experienced a power interrupt (or requests new time information) using the TIME utility.

3. Loads the BASIC interpreter.

4. Loads the program TEST43, remaining in immediate mode.

```
SET
KB2: BR 1200 DB 7 PB EVEN EXIT
TIME
BASIC
OLD "TEST43"
```

2-28. This would be a useful COMMND.SYS system command file during the development of program TEST43. Note that the words SET, EVEN, EXIT, TIME, and BASIC could have been abbreviated S, E, EX, T, and B respectively.

2-29. If the 1720A used for program development contains an optional electronic disk, the COMMND.SYS file can be expanded to speed up operation during programming. In addition to the previous tasks, this command file copies some essential system software and the program TEST43 onto the electronic disk and assigns the disk as the system device. The programmer must then remember to save the revised TEST43, after further development, onto the floppy disk.

```
SET
KB2: BR 1200 DB 7 PB EVEN EXIt
TIME
FUP
EDO:/F
YES
EDO:=FDOS.SYS,CONMON.SYS,FFUP.CIL,BASIC.CIL/B
EDO:=TEST43
EDO:/A
/X
BASIC
OLD "TEST43"
```

2-30. After the program has been developed, the working disk that uses it might contain the COMMAND.SYS system command file (described in the following paragraph). This disk automatically begins running the program TEST43 whenever the 1720A is turned on or RESTART is pressed. A Programmer Keyboard is not required unless the time clock needs to be reset.

```
TIME
BASIC
RUN "TEST43"
```

2-31. The following command file loads the BASIC interpreter, runs program TEST1, and then runs program TEST2. If program TEST1 terminates through an EXIT statement, the command file needs an additional line to reload the BASIC interpreter before attempting to run TEST2.

```
BASIC
&<ESC>[2J<ESC>[1p<ESC>[4;6HStandby for Test 2.
RUN "TEST2"
```

2-32. The message lines in the previous example use display control codes to clear the screen, select double-size characters, position the display cursor, and display a message. This is useful for a large ASCII-format program to inform the operator that Test 1 or Test 2 is being loaded. Display codes are used directly, including the escape key, without the need for quotes or the CHR$ function as in a BASIC PRINT statement. <ESC> represents an ESCape key entry.

2-33. The following command file uses FUP commands to display a directory of the system device whenever it is called and then return to the Console Monitor. If it is given the file name DIR.CMD, the system device directory can then be directly displayed from the Console Monitor by typing DIR.

```
FUP
/L
/X
```

2-34. By adding a ? character to the DIR.CMD file, it gains the ability to accept an argument. In the following example, typing DIR ED0: will produce a directory listing of the electronic disk. Typing DIR will produce a directory of the system device, as in the previous example.

```
FUP
?/L
/X
```

2-35. Creating Command Files

2-36. Two ways are presented to create a command file. Using the File Utility Program (FUP), the file can be directly typed and stored. For longer command files, the editing capabilities of BASIC are often useful. A BASIC program is presented that will generate and store a command file and a revised version of itself. Procedures for each of these methods follow:

2-37. For a short command file, use FUP to directly type the following in the file:

1. After a CONMON prompt #, type F to select the File Utility Program.

2. Type MF0: (File name)=KB0:. For the file name, use COMMND.SYS for a system command file, or (command name).CMD for a utility command file.

3. Type each command input to be stored on a separate line. Use <DELETE> to correct errors.

4. Use <ESC> directly to initiate display control sequences.

5. Use <CTRL> with the letters given in Table 2-2 to include some of the characters and control codes below character number 27 in a display string.

NOTE

If the command file requires codes below character number 27 that are not included in Table 2-2, a BASIC program must be used to generate the file.

6. Use two RETURN entries after the last line, when the last line is a display string introduced by &.

7. Terminate the file with <CTRL/Z>.

8. Type /X to return to the Console Monitor.

9. To use the command file, press RESTART on the front panel for a system command file, or type the assigned command name for a utility command file. (Do not include the .CMD extension.)

Table 2-2. Control Codes Available for FUP Generated Command Files

CONTROL CODE	CHARACTER OR FUNCTION	CONTROL CODE	CHARACTER OR FUNCTION	CONTROL CODE	CHARACTER OR FUNCTION
A	β	H	backspace	O	π
B	γ	I	horizontal tab	R	τ
D	ϵ	K	line feed	V	ψ
E	ζ	L	line feed	W	ω
F	η	N	ϕ	Y	Ω
G	beep				

2-38. For a longer command file, use the BASIC program shown in Figure 2-2. Refer to the 1720A Fluke BASIC Programming Manual for information on creating and storing BASIC programs. This program creates two files: the command file and a copy of the program with the name (command name).BAL. Note the following points:

- Use one DATA statement for each command file line.
- In place of (file name) in line 10010 use COMMND.SYS for a system command file, or (command name).CMD for a utility command file.
- In place of (file name) in lines 10120 and 10130, use COMMND.BAL for a system command file, or (command name).BAL for a utility command file.
- If the last command file line is a display string introduced by &, add one more DATA statement with a null string (DATA"").

```
1000     ON ERROR GOTO 10130
1010     DATA   "   "
1020     DATA   "   "
1030     DATA   "   "
10000    DATA "END"
10010    OPEN "(file name)" AS NEW FILE 1
10020    READ A$ \ IF A$= "END" GOTO 10110
10030    C$=""
10040    FOR I%=1% TO LEN (A$)
10050    B$=MID (A$,I%,1%)
10060    IF B$= "^" THEN I%=I%+1%\B$=CHR$(ASCII(MID (A$,I%,1%))-64%
10070    IF B$= "$" THEN B$=CHR$(27)
10080    C$=C$+B$
10090    NEXT I%
10100    PRINT #1, C$\GOTO 10020
10110    CLOSE 1
10120    KILL "(file name)"\SAVEL "(file name)"\GOTO 10140
10130    IF ERL=10120 AND ERR=305 THEN SAVEL "(file name)"ELSE OFF ERROR
10140    END
```

Figure 2-2. Command File Generator Program

- Because DATA statements cannot contain control codes such as escape (ESC), use a ^ character to indicate CTRL, and a $ character to indicate escape.

2-39. CONTROL CODES

2-40. Control character codes are formed by holding the CTRL key depressed while typing another letter, just as when using the SHIFT key. Some of these codes are intercepted by FDOS when detected and used to initiate specific actions. Table 2-3 summarizes these codes and the resulting actions.

Table 2-3. Control Code Summary

CODE	RESULT
<CTRL/C>	Stop program
<CTRL/P>	Abort program and return to CONMON
<CTRL/Q>	Continue display after stop
<CTRL/S>	Stop display
<CTRL/T>	Clear screen, reset display modes, cursor to upper left
<CTRL/U>	Delete current line
<CTRL/Z>	End-of-file code for ASCII files

Section 3
Programmer Information

3-1. INTRODUCTION

3-2. This section provides the information necessary to modify FDOS in order to perform certain specific tasks or operations (for example, modifying the IEEE-488 compatible bus handlers to accomodate instruments that are not IEEE-488 compatible). So, this section is a reference for systems programmers not a users guide. Most 1720A operations and modifications to 1720A programs can be performed in BASIC or assembly language without accessing FDOS.

CAUTION

FDOS routines do not perform error checks on their input parameters. The use of improper function codes or any illegal code may cause the system to terminate unexpectedly, or crash.

CAUTION

It is recommended that the interrupt mask not be altered under any circumstances. To do so may render the FDOS inoperable.

3-3. FDOS CALLS

3-4. Introduction

3-5. The FDOS calls device drivers directly or calls device drivers via the I/O manager (Kernel) using extended operations (XOPs). With the exception of the IEEE-488 device driver (which cannot be called via the I/O manager), device drivers are usually called via the I/O manager. The I/O manager can either be called using XOP15 or using supervisory calls (SVCs). FDOS calls have special requirements for passing arguments into memory and errors may occur when calls are executed. The following paragraphs describe XOP calls, SVC calls, passing arguments into memory and FDOS I/O manager errors.

3-6. XOP Calls

3-7. The XOP calls are software interrupts discussed in the TI Guide section of the 1720A Assembly Language Programming Manual. The TMS 9900 supports sixteen XOP calls.

0 XOP 00 through XOP 12 directly call device drivers. (A device driver is also referred to as a device handler.)

- XOP 13 and XOP 14 are available for use in any user program.

  1. Before using XOP 13, the user program must initialize the XOP 13 vector as follows:

     a. The workspace address of the XOP 13 handler must be placed at location >0074.

     b. The entry address of the XOP 13 handler must be placed at location >0076.

  2. Before using XOP 14, the user program must initialize the XOP 14 vector as follows:

     a. The workspace address of the XOP 14 handler must be placed at location >0078.

     b. The entry address of the XOP 14 handler must be placed at location >007A.

- XOP 15 calls the I/O manager.

  1. The I/O manager performs the primary operations of the FDOS.

  2. Each operation is invoked by XOP 15, but is distinguished by a different register number within the XOP call. The register numbers, operation names, and functions are listed in Table 3-1.

  3. The I/O manager can also be called using supervisory calls (SVCs).

Table 3-1. I/O Manager Functions

REGISTER NUMBER	OPERATION NAME	FUNCTION
R0	OPEN	Open a channel and associate it with a file or device
R1	CLOSE	Close a channel and make a file permanent
R2	READ	Read a block of data from a device
R3	WRITE	Write a block of data to a device
R4	DELETE	Delete a file from a file structured device
R5	RENAME	Rename a file on a file structured device
R6	LOAD	Load a memory image file (CIL) onto memory and start it

Table 3-1. I/O Manager Functions (continued)

REGISTER NUMBER	OPERATION NAME	FUNCTION
R7	EXIT	Give control back to the Operating System
R8	TIME	Return current date and time
R9	STATUS	Return information on whether or not data is available
R10	CONFIG	Tell if device configuration values will be read out of AREA or written into AREA, with changes

0 The general format for an XOP call is

```
XOP                reg,nmbr
```

1. reg is the register number (also called the argument) of the requested function.

2. nmbr is the XOP number.

3-8. The following example shows how to call the I/O manager for a Data Write.

```
XOP          R3,R15          ;WRITE A BLOCK
```

3-9. Supervisory Calls

3-10. Like XOP 15, supervisory calls (SVCs) call device handlers via the I/O manager. Unlike XOP 15, however, the SVC states the operation name which reduces the possibility of making an input error (this is a critical factor in preventing FDOS call failures). This also helps to write clear, easily understood code (especially important when a large number of calls is required).

0 The DXOP pseudo-operation and the EQUATE statement are used to set up SVC statements.

1. The following subroutine sets up SVC calls for the I/O manager to Open, Close, Read, and Write.

```
            DXOP        SVC,15        ;ASSIGN SUPERVISORY CALL
OPEN        EQU         0
CLOSE       EQU         1
READ        EQU         3
WRITE       EQU         4
```

2. A call for the I/O Manager to write a block of data can now be written directly as a Supervisory Call, containing the word WRITE, instead of as an indirect XOP call. The SVC Call is much easier to understand than a corresponding XOP call would be. This is demonstrated in the two lines of code below.

```
SVC     WRITE       ;WRITE A BLOCK USING AN SVC CALL

XOP     R3,R15      ;WRITE A BLOCK USING AN XOP CALL
```

0 Certain supervisory calls require file and device names to be passed in an argument area. The names must be supplied in RADIX-50, which is defined as a specific numerical format for storing alphanumeric characters in computer words. The procedure for placing characters into RADIX-50 format is detailed below.

1. Three alphanumeric characters can be stored in one 16-bit word.

2. RADIX-50 allows up to decimal 40 (octal 50) characters.

3. All characters are assigned a value from 0 to 39 using one of the two following methods:

   a. Select the correct value using the following rules:

CHARACTER	VALUE
space	0
A-Z	1-26
$	27
0-9	30-39

   b. Compute the correct value as follows:

      (1) Subtract 64 from the ASCII value of each alpha character.

      (2) Subtract 18 from the ASCII value of each numerical character.

4. The first character is given a numerical weight of 40*40, the second character is given a weight of 40, and the third character is given a weight of 1.

5. The weighted values of the characters are then added together to obtain the RADIX-50 equivalent

6. The assembler has a strict left-to-right priority (NOT a hierarchical one) in performing numerical operations.

   a. The expression 8+6*3 equals 42

   b. The expression 8+3*6 equals 66

   c. The expression 3*6+8 equals 26

7. For example, the RADIX-50 equivalent of MF0 is computed with the expression:

```
MF0     EQU     'M'-64*40+'F'-64*40+'0'-18  =
                77-64*40+70-64*40+48-18 = 21,070
```

NOTE

The user program must translate file and device names into RADIX-50 format. FDOS does not perform this operation.

3-11. Passing FDOS Arguments

3-12. Arguments are passed in a section of memory whose first location is called AREA.

- The contents of memory location >00A4, which is loaded during BOOTUP, act as the pointer for AREA.
- AREA is physically located in scratch-pad memory on the CPU card.
- Any assembly language routine which calls an I/O Manager (kernel) operation in FDOS first must place the arguments for that operation into AREA.
  1. The first parameter is placed in the memory location called AREA, which is pointed to by the AREA pointer.
  2. The second parameter is placed in memory location AREA+2. The second parameter's location is offset by 2 from that of the first parameter (i.e. The pointer AREA has 2 added to it to point to the next location).
  3. The third parameter is placed in memory location AREA+4.
  4. The nth parameter is placed in memory location AREA+2n.
  5. Figure 3-1 demonstrates the offset technique of loading memory. Table 3-2 lists each SVC, the arguments which must be passed to it, and the offset for each parameter.
- Each supervisory call (SVC) requires different arguments. However, the format of some of the parameters is the same for each SVC which passes that argument. The parameters with identical formats include:
  1. Channel number is an integer between 0 and 7.
  2. Length is an integer specifying the number of blocks required or allocated.

3. File reference is always passed in RADIX-50 format and consists of four parameters:

a. Device name

b. Filename 1, which contains the first three characters of a file name

c. Filename 2, which contains the last three characters of a file name

d. Extension

Table 3-2. SVC Argument Passing

SVC NAME	OFFSET	PARAMETER	DESCRIPTION
Open	(0)	Device name	
	(2)	Filename 1	
	(4)	Filename 2	
	(6)	Extension	
	(8)	Mode	0 -- old file nonzero--new file
	(10)	Channel Number	
	(12)	Required Space	if -1 -- the largest contoguous space in units of blocks if -2 -- half the largest contiguous space in units of blocks
	on return:		
	(12)	Length of file	length in blocks
	(14)	Device type	0--file structured 1--non-file structured
Close	(0)	Channel Number	
Read	(0)	Channel Number	
	(2)	Block Number	1 through max block number of file. if <0 -- next block (relative block numbers in a file are numbered from 1 to max.)
	(4)	Buffer Pointer	anywhere in memory
	(6)	Block Length	length in bytes
	on return:		
	(6)	Block Length	length of block read in bytes (Can be different from length above)

Table 3-2. SVC Argument Passing (continued)

SVC NAME	OFFSET	PARAMETER	DESCRIPTION
Write	(0)	Channel Number	
	(2)	Block Number	1 through max block number of file. if <0 -- next block (relative block numbers in a file are numbered from 1 to max.)
	(4)	Buffer Pointer	can point anywhere in memory
	(6)	Block Length	length in bytes
Delete	(0)	Device name	
	(2)	Filename 1	
	(4)	Filename 2	
	(6)	Extension	
	(8)	Channel Number	
Rename	(0)	Old device name	
	(2)	Old filename 1	
	(4)	Old filename 2	
	(6)	Old extension	
	(8)	New device name	
	(10)	New filename 1	
	(12)	New filename 2	
	(14)	New extension	
	(16)	Channel number	
Load	(0)	Device name	
	(2)	Filename 1	
	(4)	Filename 2	
	(6)	Extension	
			date format
Time	(0)	Date	[0/month/ day /year-1972] 1 5 5 5 bit bits bits bits
	(2)	Upper word of time	(10 millisecond units)
	(4)	Lower word of time	(10 millisecond units)

Table 3-2. SVC Argument Passing (continued)

SVC NAME	OFFSET	PARAMETER	DESCRIPTION
Status	(0)	Channel number	used for RS-232 devices
	on return:		users R0 contains data available status: status=0 implies no data status<>0 implies data available status is zero until at least one line of data has accumulated (i.e. until an end-of-line symbol has been read).
Config	(0)	Device name	
	(2)	Read/Write	R/W<>0 implies that a change in device configuration values will be written into AREA R/W=0 implies that the device configuration values will be read in from AREA
	(3)	baud rate	0 = 75 5 = 600 10 = 4800 1 = 110 6 = 1200 11 = 7200 2 = 134.5 7 = 1800 12 = 9600 3 = 150 8 = 2400 13 = 19200 4 = 300 9 = 3600
	(4)	uart control	universal asynchronous receive/transmit -- described in "TI 9900 Family Systems Design" manual
	(5)	end of line value	
	(6)	end of file value	
Exit	-	No parameters	

3-13. FDOS I/O Manager Errors

3-14. FDOS may encounter an error during the execution of an I/O Manager operation. (For example, a floppy disk could be write protected when a WRITE SVC is called.)

- If an error is encountered while executing an SVC call, the program continues executing at the first word following the call.
- If no error is encountered, the program skips the first word after the call and continues at the next location.

MEMORY LOCATION	CONTENTS
>004A	pointer to AREA
AREA	parameter, offset is 0
AREA+2	parameter, offset is 2
AREA+4	parameter, offset is 4
.	. . .
.	. . .
.	. . .
AREA+2n	parameter, offset is 2n

For the SVC call LOAD, the offset for each parameter is

PARAMETER	OFFSET
device name	0
filename 1	2
filename 2	4
extension	6

Figure 3-1. AREA Offsets

0 Whenever FDOS encounters an error, it passes an error number (listed in Table 3-3) in Register R0 of the calling workspace.

1. The user program should contain an error exit, such as a JUMP statement, in the first location following the SVC. It should exit the Call subroutine and jump to an error display subroutine. Failure to exit the SVC call when an error is detected may lead to a sudden termination of system operations.

2. The error numbers passed in Register R0 are NOT displayed or listed. A user subroutine must be written to take recovery action, such as printing an error message.

3-15. The following example shows a program that assigns supervisory calls, moves the Area Pointer from location >00A4 to Register R3, places each parameter into the location pointed to by the contents of R3 (with the appropriate offset), and performs the READ operation.

EXAMPLE:

```
      DXOP        SVC,15              ;ASSIGN SUPERVISORY CALL
READ  EQU         2
AREA  EQU         >A4

(1)  START  MOV         @AREA,R3      ;GET POINTER TO ARGUMENT AREA
(2)         CLR         *R3+          ;SET UP CHANNEL NUMBER 0
(3)         LI          R4,31         ;SET UP THE
(4)         MOV         R4,*R3+           BLOCK NUMBER
(5)         LI          R4,BUFFER     ;SET UP THE POINTER
(6)         MOV         R4,*R3+           TO THE READ BUFFER
(7)         LI          R4,512        ;SET UP THE BLOCK
(8)         MOV         R4,*R3            LENGTH (512 BYTES)
(9)         SVC         READ          ;PERFORM THE READ OPERATION
(10)        JMP         ERROR         ;ERROR EXIT
(11)        ***                       ;CONTINUE ON
```

Explanation:

(1) R3 will point to the first element of the argument area.

(2) The first element is the channel number, which will be cleared in order to indicate channel 0.

(3) The second element is the block number, which is set to 31 in order to access the thirty-first block of a previously opened file.

(5) The third element must point to where the block is to be loaded.

(7) The fourth element indicates how many bytes are going to be read.

(9) This is the SVC call to FDOS.

(10) If there is an error, the program jumps to the subroutine labeled ERROR.

(11) The program comes here if the read was successful.

Table 3-3. FDOS Execution Errors

ERROR NO.	MEANING
0	No error
2	Device hardware error
4	Device not ready
6	Medium write protected
8	Illegal channel number
10	Channel in use
12	Not a valid device name
14	File not found
16	No room on medium
18	Attempt to read/write past end of file
20	Channel not open
22	Devices do not match in 'RENAME'
24	Non-existent memory
26	Non-file structured device interrupt buffer overflow
28	Medium was changed
30	Illegal directory

3-16. DIRECT CALLS TO DEVICE HANDLERS

3-17. Introduction

3-18. Each device on the system has a unique handler which provides an interface between the FDOS and the hardware. Five of the 13 XOP levels (0 through 12) which are allocated to the device handlers are in use: 0, 1, 2, 3, 4, 5, and 12. The mnemonic and XOP number associated with each of the five device handlers are listed in Table 3-4. The I/O manager handles all of the devices except GPIB, so only direct calls to the KB0 and GPIB handlers are recommended. KB0 handles the keyboard and the video display. Since the I/O manager does not handle the GPIB device, GPIB must be called directly.

3-19. XOP Format

3-20. With the exception of GPIB, the XOPs which pass information to and from a handler device all have the following format:

```
XOP          pntr,nmbr
```

- nmbr represents the particular XOP number.
- pntr is a pointer that points to the data area that contains the parameters and has the format:
  1. The first word is the absolute block number.
  2. The second word is the buffer address.
  3. The third word is two bytes. The upper byte is the function code and the lower byte is not used.
  4. The fourth word is the record length in bytes.
  5. The fifth word is the error code.
- These parameters can be passed in both directions.

Table 3-4. XOP Numbers of Device Handlers

DEVICE	MNEMONIC	XOP NUMBERS
Floppy Disk	MF0	0
Keyboard & Video	KB0	1
RS-232 Serial Port #1	KB1	3
RS-232 Serial Port #2	KB2	4
Electronic Disk	ED0	5
IEEE-488 Bus Ports	GPIB	12

3-21. DEVICE HANDLER INTERFACES

3-22. Introduction

3-23. The following paragraphs describe the interfaces to the five handlers.

3-24. Minifloppy Disk Handler (MF0)

3-25. The MF0 device handler has the following characteristics:

- MF0 is called by XOP number 0.

- MF0 provides the interface to the 1720A Instrument Controller's disk hardware.

- MF0 Error codes are:

ERROR NO	DESCRIPTION
0	No error
2	Hard diskette error
4	Diskette not loaded
6	Diskette write protected
24	Non-existent memory
28	Diskette was swapped

- MF0 Function codes are defined as follows:

NUMBER	NAME	ACTIONS
0	Restore Head	The floppy disk read/write head is moved to a position over track 0.
2	Read a Block	N bytes of data (record length) are transferred from the floppy diskette to main memory (buffer address). Prior to the start of the transfer the read/write head is positioned at the specified block (absolute block number).
4	Write a Block	N bytes of data (record length) are transferred from memory (buffer length) are transferred from memory (buffer address) to the diskette on MF0. Prior to the start of the transfer the read/write head is positioned at the specified block (absolute block number).
6	Test Medium Swapped	Test a bit in the floppy interface to determine if the diskette in Drive 0 has been removed and replaced (swapped) since the diskette was swapped. This function is executed prior to accessing a memory resident it describes is still mounted.
8	Step In	Move read/write head to next higher numbered track.
10	Write Track	One full track, both data and format information, is written to the diskette on the designated drive (MF0: or MF1:). Both data and format bytes must appear in the designated buffer (buffer address). See Appendix A of this manual for a description of the format bytes.

3-26. Video/Keyboard Handler (KB0)

3-27. The KB0 device handler has the following characteristics:

- KB0 is called by XOP number 1.
- The video screen and the keyboard together are the primary user-system interface. The handler determines many of the characteristics of that interface.
- KB0 has no Error codes, but does have an error exit for Function code 2 (Read a Record) and Function code 4 (Write a Record).
- KB0 Function codes are as follows:

NUMBER	NAME	ACTIONS
0	Read a Line	Returns a line from the console or a command file starting at the locations pointed to by the buffer pointer. No length is returned.
2	Read a Record	Returns a line (= record) from the console or a command file. The length of the record is returned.
4	Write a Record	Writes a record to the console terminals. Writing is complete when the record length is reached, an end-of-file character is encountered, or when a CTRL/C is given.
6	Read a Character	Returns a character from the console terminal or the command file in R8 of the calling workspace.
8	Write a Character	Writes a character which is in R8 of the calling workspace to the console terminal.
10	Set Input Mode	If the value of the buffer pointer is 0, then the line mode is entered. Otherwise, the driver will go into the single character mode.
12	Write a Line	Writes a line to the screen which is pointed to by the buffer pointer. The last character must be negated. When a command file is active, this function is ignored.
14	No Action	Used by other non-file structured devices to reset their queues.
16	Return Status	Checks to see is at least one line (in Line Mode) or character (in Character Mode) is available for input. If no line or character is available, R0 (the user's register) is set to zero. Otherwise the number of line(s) or character(s) is returned.

NUMBER	NAME	ACTIONS
18	Configure	Checks the R/W flag. If a read operation is requested, the baud rate protion of the port configuration is returned in AREA. If a write operation is requested, only the baud rate of KB0 is updated.

0 The video/keyboard handler can operate in the Line Mode or in the Character Mode.

1. Line Mode

   a. When KB0 is in the Line Mode, a line is not returned to the display until KB0 receives a line terminator (line terminator can be <cr>, CTRL/Z, or CTRL/T).

   b. The following control key operations are active:

      (1) CRTL/U which deletes the current line. CTRL/U echoes (displayed on the screen) as ^U<cr><lf>.

      (2) CTRL/C which deletes all lines in buffer and sets the control/C flag (CC) to a value of 3. CTRL/C echoes as ^C<cr><lf>.

      (3) CTRL/P which deletes all lines in buffer, terminates the current programs and command file (if active), closes all files, and gives control back to CONMON. CTRL/P echoes as ^P<cr><lf>.

      (4) CTRL/S which stalls the output to the display. CTRL/S does not echo.

      (5) CTRL/Q which restarts the stalled output to the display. CTRL/Q does not echo.

      (6) CTRL/Z which terminates the current line. CTRL/Z acts as the End-of-file and echoes as ^Z<cr><lf>.

      (7) CTRL/T which terminates the current line. CTRL/T echoes as <ESC>[;2;4p]<cr><lf>, which clears the screen and sets the screen to no graphics, small size, and no keyboard lock.

      (8) Any control key operations not listed are stored in the input buffer and are echoed as a >7F character, which is an inverse video block.

2. Character Mode

   a. The display handler can accept a signal from the console display (and input buffer) without requiring a line terminator. Echoing is disabled.

   b. The keyboard entries <DELETE>, CTRL/U, CTRL/Z, CTRL/P, CTRL/C, CTRL/T, and <RETURN> are not active, but are returned as are all other entries. CTRL/C and CTRL/P also store their respective values in CC (CTRL/C = 3 and CTRL/P = 16).

0 The operation of the Video/keyboard handler is affected by the status of the Command File Processor (see Appendix A).

3-28. The first of the following examples shows a program that prints a line of test to the console device in Line mode. The second of the following examples shows a program that puts KB0 into the Character Mode and accepts a single character from the keyboard.

EXAMPLE 1:

The program is as follows:

```
(1)           BL      @PRINT          ;PRINT MESSAGE
(2)           DATA    T1
      *
(3)  T1       TEXT    -'This is a message'
(4)           DXOP    KB0,1           ;ASSIGN KEYBOARD DRIVER
      *                                   CALL
(5)  PRINT    EQU     $               ;SUBROUTINE ENTRY POINT
(6)           MOV     *R11+,@PP+2     ;STORE TEXT POINTER IN
      *                                   ARGUMENT LIST
(7)           KB0     @PP             ;PRINT THE LINE
(8)           RT                      ;RETURN
(9)  PP       EQU     $-2             ;ARGUMENT LIST
(10)          BSS     2               ;EMPTY SPOT FOR TEXT
      *                                   POINTER
(11)          BYTE    12              ;FUNCTION CODE
(12)          EVEN
```

The explanation of the program is as follows:

LINE — EXPLANATION

(1) This is an example of the use of the subroutine PRINT.

(2) The DATA statement following the subroutine call must point to the beginning of the text to be printed

(3) This is a typical text statement. Note the minus sign, which causes the last byte to be negated so the driver knows when to terminate printing.

(4) This allows a symbolic call to the driver.

(5) This is the required print subroutine.

(6) The word following the call is stored in the second word of the argument list PP.

(7) The call is made here, with a pointer to the argument list.

(8) Return to caller.

(9) This is the start of the argument list. Since the first word is not used with this function code, the label of the argument is set one word backwards.

(10) The pointer to the text is written in this word.

(11) This function code tells the driver to print one line of text.

(12) This directive causes the location counter to start on an even boundary again.

EXAMPLE 2:

The program is as follows:

```
(1)              KB0       @SM              ;GO INTO SINGLE CHARACTER
                                                MODE
(2)              KB0       @ACCEPT          ;ACCEPT A CHARACTER
                 ***
(3)  SM          EQU       $-2              ;SINGLE CHARACTER MODE
(4)              DATA      -1               ;SET BOOLEAN TO NON ZERO
(5)              BYTE      10               ;FUNCTION CODE
                 EVEN
(6)  ACCEPT      EQU       $-4              ;ACCEPT A CHARACTER
(7)              BYTE      6                ;FUNCTION CODE
                 EVEN
```

The explanation is as follows:

LINE EXPLANATION

(1) This call tells the console driver to accept single characters from now on.

(2) This call requests a single character from the console keyboard. The character is returned in R8.

(3) This points to the beginning of the argument list. Only two words are necessary.

(4) This location acts as a boolean, if non-zero character mode is indicated.

(5) Function code 10.

(6) This points to the beginning of the argument list. Only one word is needed.

(7) Function code 6.

3-29. RS-232 Port Handlers (KB1/KB2)

3-30. The KB1 and KB2 device handlers have the following characteristics:

- KB1 is called by XOP number 3.
- KB2 is called by XOP number 4.
- KB1 and KB2 are two identical RS-232 ports
- The error code for KB1 and KB2 is:

ERROR NO	DESCRIPTION
26	Lost data on input

- KB1 and KB2 Function codes are defined as follows:

NUMBER	NAME	ACTIONS
2	Read a Record	Returns a record (line) from the specified port ('KB1'/'KB2') when a line feed character is input or when 512 characters have been input. The record will also be return if a CTRL/C is seen on the console terminal. If the 132-character input interrupt buffer is full when a character is ready to be input, an error 26 occurs ('lost data on input'). The record length is returned.
4	Write a Record	A record is written to the specified port ('KB1'/'KB2'). Writing is complete when the record length is reached, when a CTRL/C is seen on the console terminal, or when an End-of-File character is encountered.
14	Reset Ring Buffers	The interrupt ring buffers are cleared and the lost data flag is cleared. This function is executed by the OPEN XOP handler.

NUMBER	NAME	ACTIONS
16	Return Status	A check is made to see if at least one line of data is available for input. If no lines are available, the user's register R0 is set to zero. Otherwise, the number of lines available is returned in R0.
18	Configure	The R/W flag is checked. If a read operation is requested, the currect configuration of the port is returned in AREA. If a write operation is requested, the information in AREA is used to modify the port's configuration.

3-31. **E-Disk Handler (EDO)**

3-32. The EDO device handler has the following characteristics:

- EDO is called by XOP number 5.
- EDO is the interface to the electronic disk memory board.
- EDO Error codes are:

ERROR NO	DESCRIPTION
0	No Error
2	Parity Error

- EDO Function codes are defined as follows:

NUMBER	NAME	ACTIONS
2	Read a Block	'N' bytes of data ('record length') are transferred from the E-Disk to main memory ('buffer address'). The Transfer is from the specified block ('absolute block number').
4	Write a Block	'N' bytes of data ('record length') are transferred from main memory ('buffer address') to the E-Disk at a specified block ('absolute block number').

3-33. **IEEE-488 Bus Handler (GPIB)**

3-34. The IEEE-488 Bus handler has a different structure than the other device handlers. The primary differences include:

- The parameters passed to the handler
- The function codes
- The error codes.

3-35. The GPIB (General Purpose Interface Bus) handler can handle two IEEE-488 standard interface compatible boards with two ports each.

- The GPIB handler is accessed via the XOP call

```
XOP                    pntr,12
```

  1. pntr is the pointer which points to the data area containing the parameters.

  2. 12 is the XOP number.

- The GPIB handler is accessed directly, rather than going through the I/O Manager as the other handlers do.

- The four ports are numbered from 0 through 3.

- The format of the data area to which the pointer refers is as follows:

```
first word:                    addressed ports mask
second word:                   pointer to device list
third word:
a. upper byte:                       unused
b. lower byte:                       function code
fourth word:                   unused
fifth word:                    error return code
```

  1. The first word is the addressed ports mask.

     a. In the addressed ports mask, the bit settings denote different ports.

     b. The mask addresses port number n when the 2^n bit is set in the mask (the notation 2^n means the nth bit).

     c. A null mask contains all zeros.

     d. The addressed ports mask is usually referred to as the port mask.

  2. The second word is the pointer to device list.

     a. A pointer device list is used for commands which require that a device or set of devices be addressed (such as the commands in a PRINT routine).

     b. A sequence of one-byte integers is used to address the particular port, the primary address, and any secondary addresses required.

     c. A secondary address is distinguished from the primary address by having the 2^6 bit in the address word set.

d. The end of the list is designated by having the 2^7 bit (the SIGN bit) set.

e. The end-of-list pointer is not used for addressing. It means that the last entry has already been processed not that this is the last entry.

f. The format of the device list to which the pointer points to is shown in Figure 3-2.

3. The upper byte of the third word contains the function codes.

a. For each specific call to the GPIB handler, a function code specifies what operation the handler is to perform.

b. The following list provides the function code number for each of the 24 GPIB function codes, describes the function, and refers to a figure in which the function is described in detail.

FUNCTION CODE NUMBER	DESCRIPTION	FIGURE NUMBER
0	Initialize bus	5-3
2	Set REN line	5-4
4	Reset REN line (local)	5-5
6	Send GO TO LOCAL command	5-6
8	Address a group of listeners	5-7
10	Address a talker	5-8
12	Send DEVICE CLEAR	5-9
14	Send SELECTED DEVICE CLEAR	5-10
16	Send LOCAL LOCKOUT	5-11
18	Send GROUP EXECUTE TRIGGER	5-12
20	Send PARALLEL POLL CONFIGURE	5-13
22	Send PARALLEL POLL DISABLE	5-14
24	Read binary data	5-15
26	Write binary data	5-16
28	Input a line of data	5-17

FUNCTION CODE NUMBER	DESCRIPTION	FIGURE NUMBER
30	Print a line of data	5-18
32	Set timeout limit	5-19
34	Set INPUT terminator	5-20
36	Return active SRQ mask	5-21
38	Update and return active SRQ mask	5-22
40	Perform serial poll	5-23
42	Perform parallel poll	5-24
44	Send INTERFACE CLEAR	5-25
46	Re-enable SRQ interrupts on port	5-26

4. The fifth word contains the error return codes. If an error is encountered during processing, an error code is returned in the last parameter location of the function code. The error codes are as follows:

ERROR CODE NO.	DESCRIPTION
2	Input buffer overflow
4	Illegal function code
6	Handshake not completed
8	Too may ports specified
10	No devices attached to port
12	No ports available
14	Illegal or unavailable port specified
16	BUS I/O timeout
18	EOI and ATN specified to be output with a WBYTE command

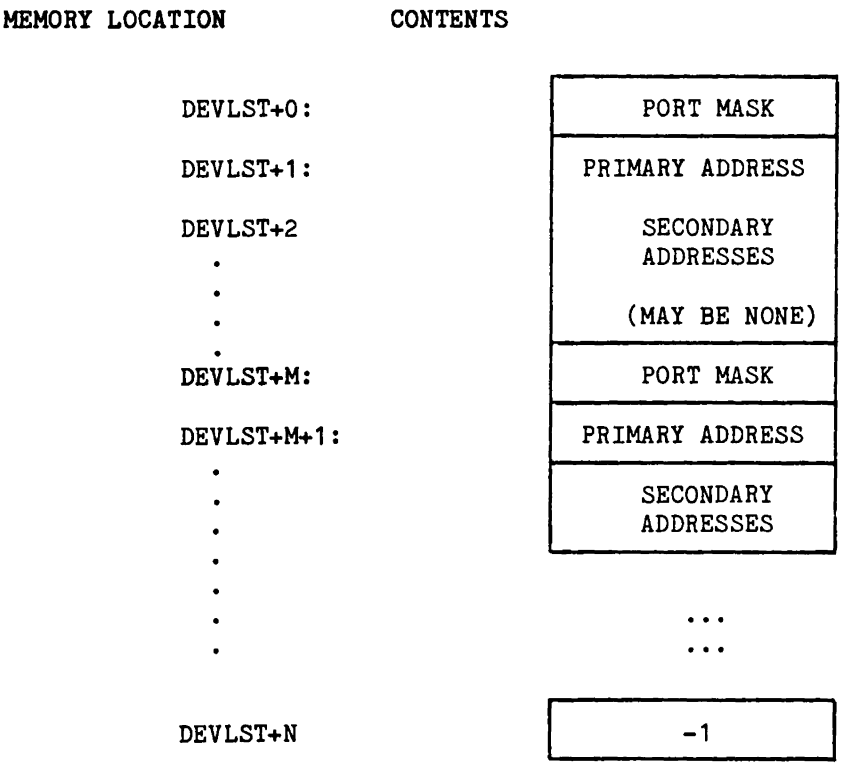


Figure 3-2. Device List Format

FUNCTION CODE NAME: Initialize Bus

FUNCTION CODE NUMBER: 0

DESCRIPTION: Sets the 'IFC' bus line TRUE for 100 microseconds (100 usec) on the designated ports. It then sends a PPU (Parallel Poll Unconfigure) an UNL (Unlisten), and an UNT (Untalk) on the bus.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARSM+2:	unused
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS:

REN
-ATN
IFC
-IFC (after 100 usec delay)
ATN
UNL
UNT
PPU

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No devices attached to port
16	Bus timeout

FIGURE 3-3. Initialize Bus Function Code

FUNCTION CODE NAME: Remote

FUNCTION CODE NUMBER: 2

DESCRIPTION: Sets the Remote Enable line (REN) true on the designated ports.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Unused
PARMS+4:	Function Code
PARMS+6:	Unused
PARMS+8:	Error Return Word

BUS SIGNALS: REN

ERRORS RETURNED: NONE

Figure 3-4. Remote Function Code

FUNCTION CODE NAME: Local

FUNCTION CODE NUMBER: 4

DESCRIPTION: Causes the Remote Enable line (REN) on the designated ports to be reset. This is done only after any active transactions on the bus are completed. If they are not completed in the appropriate time, a Timeout error occurs.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	unused
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS: -REN (after completion of last handshake)

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
16	Bus Timeout

Figure 3-5. Local Function Code

FUNCTION CODE NAME: Send GO-TO-LOCAL

FUNCTION CODE NUMBER: 6

DESCRIPTION: Addresses a set of devices as listeners and then sends the Go-To-Local (GTL) command on the bus.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Pointer to Devlist
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS:
ATN
UNL
UNT
MLA (For each device addressed)
GTL

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No device attached to port
16	Bus timeout

Figure 3-6. Send GO-TO-LOCAL Function Code

FUNCTION CODE NAME: Address Listeners

FUNCTION CODE NUMBER: 8

DESCRIPTION: Sends a listen address (MLA) to the devices specified in the device list pointed to by the pointer in location PARMS+2.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Pointer to Devlst
PARMS+2:	unused
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS:
ATN
UNL
UNT
MLA (For each device addressed)

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No devices attached to port
16	Bus Timeout

Figure 3-7. Address Listeners Function Code

FUNCTION CODE NAME: **Address Talker**

FUNCTION CODE NUMBER: 10

DESCRIPTION: Sends a talk address (MTA) to the device specified in the XOP Call.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Pointer to Devlst
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNAL:
ATN
UNL
UNT
MTA

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No device attached to port
16	Bus Timeout

Figure 3-8. Address Talker Function Code

FUNCTION CODE NAME: Device Clear

FUNCTION CODE NUMBER: 12

DESCRIPTION: Sends a Device Clear (DCL) command to all ports addressed by the call.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	unused
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS:
ATN
UNL
UNT
DCL

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No devices attached to port
16	Bus timeout

Figure 3-9. Device Clear Function Code

FUNCTION CODE NAME: Send Selected Device Clear

FUNCTION CODE NUMBER: 14

DESCRIPTION: Addresses (as listeners) the set of devices specified by the device list. It then sends a Selected Device Clear (SDC) command to the ports on which the listeners reside.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Pointer to Devlst
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS:
ATN
UNL
UNT
MLA (For each device in list)
SDC
UNL

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No devices attached to port
16	Bus timeout

Figure 3-10. Send Selected Device Clear Function Code

FUNCTION CODE NAME: Local Lockout

FUNCTION CODE NUMBER: 16

DESCRIPTION: Sends the Local Lockout (LLO) message to the ports addressed by the call, after 'REN' is set true.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	unused
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS: **REN**
ATN
LLO

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No devices attached to port
16	Bus timeout

Figure 3-11. Local Lockout Function Code

FUNCTION CODE NAME: Send Group Execute Trigger

FUNCTION CODE NUMBER: 18

DESCRIPTION: Sends a listen address (MLA) to the set of devices specified in the device list pointed to in the XOP call. The Group Execute Trigger MEssage (GET) is then sent to the ports on which the listeners reside.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Pointer to Devlst
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS:
ATN
UNL
UNT
MLA (For each device in DEVLST)
GET

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No device attached to port
16	Bus timeout

Figure 3-12. Send Group Execute Trigger Function Code

FUNCTION CODE NAME: Send Parallel Poll Configure

FUNCTION CODE NUMBER: 20

DESCRIPTION: Sends a listen address (MLA) to the device specified in the device list pointed to in the XOP Call. The device is then configured as specified in call parameter PARMS+6. The SENSE in the low byte is either 0 or 1, while the BIT NO. in the high byte may range from 0 to 7, indicating the DIO line for the response to a parallel poll. The SENSE and the BIT NO. parameters are right-justified in the low and the high bytes, respectively of the Call word.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Pointer to Devlst
PARMS+4:	Function Code
PARMS+6:	Bit No. ¦ Sense
PARMS+8:	Error Return Word

BUS SIGNALS:
ATN
UNL
UNT
MLA
PPC
PPE (ORed with 'BIT NO.' and 'SENSE')
UNL

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No devices attached to port
16	Bus timeout

Figure 3-13. Send Parallel Poll Configure Function Code

FUNCTION CODE NAME: Send Parallel Poll Disable

FUNCTION CODE NUMBER: 22

DESCRIPTION: Sends a listen address (MLA) to the device specified on the device list pointed to in the XOP call. A Parallel Poll Disable (PPD) command is then sent on that port. Only one port may be specified.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Pointer to Devlst
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS:
ATN
UNL
UNT
MLA
PPD
UNL

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No devices attached to port
16	Bus timeout

Figure 3-14. Send Parallel Poll Disable Function Code

FUNCTION CODE NAME: Read Binary Data

FUNCTION CODE NUMBER: 24

DESCRIPTION: Moves data from the bus to either word arrays or byte arrays in memory.

0 If the 'OUTPUT FIELD FLAG' is zero, word-aligned output is to be generated. When the data is placed in a word array, the low order byte contains the value and the high order byte indicates whether or not EOI was asserted with that data byte. If EOI was asserted, the 2^8 bit of that word will be set.

0 If the 'OUTPUT FIELD FLAG' is nonzero, byte-aligned output is to be generated. No EOI data is available with the byte array.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Pointer to Array
PARMS+4:	Function Code
PARMS+6:	Number of Bytes to Read
PARMS+8:	Error Return Word
PARMS+10:	Output Field Flag

BUS SIGNALS: NONE -- The array is filled with data bytes.

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No devices attached to port
16	Bus timeout

Figure 3-15. Read Binary Data Function Code

FUNCTION CODE NAME: Write Integer Array

FUNCTION CODE NUMBER: 26

DESCRIPTION: Sends the contents of the word-aligned array pointed to in the call parameters onto the bus. The 'EOI' line is set if the 2^8 bit is set; the 'ATN' line is set if the 2^9 bit is set. However, an error is returned if both 'EOI' and 'ATN' are to be set. Each data byte should be right-justified in an array word. Only one port may be addressed by this command.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Pointer to Array
PARMS+4:	Function Code
PARMS+6:	Number of Bytes to Send
PARMS+8:	Error Return Word

BUS SIGNALS: EOI
ATN

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
8	More than one port specified
16	Bus timeout
18	EOI and ATN both flagged

Figure 3-16. Write Integer Array Function Code

FUNCTION CODE NAME: Input A Line

FUNCTION CODE NUMBER: 28

DESCRIPTION: Moves data from the bus into the buffer pointed to by the pointer. The function is terminated when an EOI is detected, or when an input termination character is received (see Function Code 34). The number of data bytes actually placed in the buffer is returned.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Pointer to Buffer
PARMS+4:	Function Code
PARMS+6:	Buffer Length (bytes)
PARMS+8:	Error Return Word

OUTPUT: PARMS+6 Word --- Count of characters placed in the buffer.

BUS SIGNALS: Data Received

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
2	Input buffer overflow
10	No deviced attached to port
16	Bus timeout

Figure 3-17. Input A Line Function Code

FUNCTION CODE NAME: Print A Line

FUNCTION CODE NUMBER: 30

DESCRIPTION: Sends the contents of a byte-aligned array pointed to by the pointer onto the bus.

- PARMS+6 gives the number of bytes to be sent
- PARMS+10 designates the order in which data is stored in the buffer:
  1. A value greater than or equal to zero indicates that succeeding data bytes are stored in successively higher memory locations (normal order)
  2. A value less than zero indicates that succeeding data bytes are stored in lower memory locations (reverse order)
- PARMS+12 is set nonzero if EOI is to be asserted true when the last byte of data in the buffer is sent.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Buffer Address (pointer)
PARMS+4:	Function Code
PARMS+6:	Number of Bytes to Send
PARMS+8:	Error Return Word
PARMS+10:	Order of Data in Buffer
PARMS+12:	EOI on Last Byte Flag

BUS SIGNALS: The data is sent to the bus. EOI is sent with the last data byte if requested.

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
16	Bus timeout

Figure 3-18. Print A Line Function Code

FUNCTION CODE NAME: Set Timeout Limit

FUNCTION CODE NUMBER: 32

DESCRIPTION:

0 Takes the time limit parameter in milliseconds (msec), converts it to 10-msec ticks by truncation, and stores it in the timer limit word. The time limit parameter (PARMS+2) should be given in msec.

0 A timeout limit of zero (0) disables any check for bus timeout.

0 The maximum time limit is 32767 milliseconds.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Timeout Limit
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS: None -- TIMLMT is set in 10 msec ticks

ERRORS RETURNED: None

Figure 3-19. Set Timeout Limit Function Code

FUNCTION CODE NAME: Set Terminating Character

FUNCTION CODE NUMBER: 34

DESCRIPTION: Sets the character to be used as the delimiter for an input string (as read by Function Code 28), or can indicate that EOI is to be the only terminator. In PARMS+2, Terminator is the terminating character to be used. Flag indicates that the current terminator, or that no terminator should be used.

0 Flag = 0: Replace current terminator with new one.

0 Flag <>0: Use no terminator except 'EOI' assertion.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	unused
PARMS+2:	Terminator ¦ Flag
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS: None

ERRORS RETURNED: None

Figure 3-20. Set Terminating Character Function Code

FUNCTION CODE NAME: Return SRQ Status

FUNCTION CODE NUMBER: 38

DESCRIPTION: Similar to the Test SRQ Status (Function Code 36), except that Return SRQ Status does not cause a test of the ports. It only returns current status as reflected by the occurrence of past SRQ interrupts from the active GPIB ports.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	unused
PARMS+2:	unused
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

OUTPUT: PARMS+2, which contains the port mask for ports which have active service requests.

BUS SIGNALS: None

ERRORS RETURNED: None

Figure 3-22. Return SRQ Status Function Code

FUNCTION CODE NAME: Serial Poll

FUNCTION CODE NUMBER: 40

DESCRIPTION: Performs a serial poll on the device in the device list pointed to by PARMS+2, by addressing the device as a talker, sending Serial Poll Enable (SPE), and recieving the instrument response. The response is returned to the caller in PARMS+2, in a right-justified, zero-filled to left format.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	Pointer to DEVLST
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS:

ATN
UNL
UNT
MTA
SPE
-ATN
(Accepts Status Byte from Device)
ATN
UNT
SPD

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No devices attached to port
16	Bus timeout

Figure 3-23. Serial Poll Function Code

FUNCTION CODE NAME: Parallel Poll

FUNCTION CODE NUMBER: 42

DESCRIPTION: Performs a parallel poll on the port specified in the XOP call. The response is returned to the caller in PARMS+2 in the low byte, with the high byte zeroed.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	unused
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS:
ATN
EOI
(Accepts Poll data from Port)
-EOI

ERRORS RETURNED:

ERROR NUMBER	DESCRIPTION
10	No devices attached to port
16	Bus timeout

Figure 3-24. Parallel Poll Function Code

FUNCTION CODE NAME: Interface Clear

FUNCTION CODE NUMBER: 44

DESCRIPTION: Sets the Interface Clear (IFC) message, on the designated ports for 100 microseconds. No check is made for ongoing bus I/O before 'IFC' is asserted.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	unused
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS: IFC
(Wait for at least 100 Microseconds)
-IFC

ERRORS RETURNED: None

Figure 3-25. Interface Clear Function Code

FUNCTION CODE NAME: Re-enable SRQ Interrupt

FUNCTION CODE NUMBER: 46

DESCRIPTION: Enables the SRQ interrupts on the ports designated by the port mask. Interrupts on the ports are disabled when SRQ is received. This function also resets the "SRQ Received" status for the ports designated in the call.

INPUT PARAMETERS:

LOCATION	CONTENTS
PARMS+0:	Port Mask
PARMS+2:	unused
PARMS+4:	Function Code
PARMS+6:	unused
PARMS+8:	Error Return Word

BUS SIGNALS: None

ERRORS RETURNED: None

Figure 3-26. Re-enable SRQ Interrupt Function Code

Appendix A
1720A Instrument Controller Disk File Structure

A-1. INTRODUCTION

A-2. The information in this appendix summarizes the structure of data and programs on floppy disks as implemented on the John Fluke Model 1720A Instrument Controller.

A-3. FILE-STRUCTURED DEVICE ORGANIZATION

A-4. A file-structured device is usually built out of blocks. Each block is 512 bytes in length. When the disk is zeroed, the File Utility Program (FUP) initializes the directory with the number of available blocks. The first 2 blocks are used for the directory. The remaining space is available for either program files or data files.

A-5. DIRECTORY LAYOUT

A-6. The length of the directory is expressed in segments. Each segment is two blocks. The directory consists of a header and a number of directory entries. Table A-1 lists contenst of the five directory header words and Table A-2 lists the contents of the seven directory segment words.

Table A-1. Directory Header

HEADER WORD	CONTENTS
First Word	Number of avaiable directory segments. Each segment is two blocks long.
Second Word	Current segment number. Currently set to 1.
Third Word	Not used.
Fourth Word	Number of extra words per entry.
Fifth Word	First available block number.

Table A-2. Directory Segment

SEGMENT WORD	CONTENTS
Word 0	Entry Status: 0 >100 is a tentative entry. A file has been opened, but not yet closed. 0 >200 is an empty entry. 0 >400 is a permanent entry. The file is a permanent file. 0 >800 is an end of directory segment.
Word 1	The first three characters of the file name written in Radix-50 notation.
Word 2	The last three characters of the file name written in Radix-50 notation.
Word 3	Extension written in Radix-50 notation.
Word 4	The length of the file in blocks.
Word 5	The channel number associated with a file is stored at this location when a file is tentatively opened as a new file.
Word 6	The date the file was created. If Word 6 is a zero, no date was available when the file was created. The date format is as follows: Bits 0-3 Year minus 1972 Bits 4-9 Day (1 to 31) Bits 10-14 Month (1 to 12) Bit 15 Zero

A-7. RADIX-50 FORMAT

A-8. RADIX-50 Format is descried in detail in Section 5 of this manual. The following material breifly describes RADIX-50 Format.

A-9. In RADIX-50 Format, three alphanumeric characters can be stored in only 16 bits by limiting the number of different characters to 40 (50 octal). Use the following procedure to find the RADIX-50 equivalent of three alphanumeric characters:

0 Assign the correct value to each alphanumeric character one of two ways:

1. Select the appropriate value for each character from Table A-3.

2. Compute the correct value as follows:

   a. Subtract 64 from the ASCII value of each alpha character.

   b. Subtract 18 from the ASCII value of each numerical character.

- The first character is given a numerical weight of 40*40, the second character is given a weight of 40, and the third character is given a weight of 1.

- The weighted values of the characters are then added together to obtain the RADIX-50 equivalent

- The assembler has a strict left-to-right priority (NOT a hierarchical one) in performing numerical operations.

  1. The expression 8+6*3 equals 42.

  2. The expression 8+3*6 equals 66.

  3. The expression 3*6+8 equals 26.

A-10. The following example shows how to find the RADIX-50 equivalent of MF0.

```
MF0     EQU     'M'-64*40+'F'-64*40+'0'-18 =
                77-66*40+70-64*40+48-18 = 21,070
```

NOTE

The user program must translate file and device names into RADIX-50 format. FDOS does not perform this operation.

Table A-3. RADIX-50 Format Value Allocation

CHARACTER	NUMBER	CHARACTER	NUMBER	CHARACTER	NUMBER
Space	0	M	13	Z	26
A	1	N	14	$	27
B	2	O	15	0	30
C	3	P	16	1	31
D	4	Q	17	2	32
E	5	R	18	3	33
F	6	S	19	4	34
G	7	T	20	5	35
H	8	U	21	6	36
I	9	V	22	7	37
J	10	W	23	8	38
K	11	X	24	9	39
L	12	Y	25		

A-10. MODIFIED IBM FORMAT

A-11. The 1720A uases a modified IBM format: MFM, 10 sectors per track, 1 block = 1 sector, and 512 bytes per sector.

NUMBER OF BYTES		HEX VALUE OF BYTE
80		4E
12		00
3		F6
1		FC (Index Mark)
16		4E
14		00
3		F5
1		FE (ID Address Mark)
1		Track Number (0 through 39)
1		Side Number (0 or 1)
1		Sector Number (1 through 10)
1		02
1	*	F7 (2 CRCs written)
22		4E
12		00
3		F5
1		FB (Data Address Mark)
512		DATA
1		F7 (2 CRCs written)
22		4E
598**		4E

* Write the bracketed field 10 times.

** Continue writing until FD 1791 interrupts out (approximately 598 bytes).

A-12. INTERLEAVING

A-13. The sectors in the disk are interleaved in order to increase throughput. Interleaving is accomplished by giving the sectors a number in their ID field which differes from the physical location. Sectors are also skewed from track to track to compensate for step times. Table A-4 shows the relationship between physical and logical sector numbering.

NOTE

The interleaving and skewing is not a requirement for the 1720A to operate properly so sectors can be numbered as they appear in physical order

Table A-4. Physical Versus Logical Numbering

TRACK	PHYSICAL SECTOR NUMBER									
	1	2	3	4	5	6	7	8	9	10
0	1	6	2	7	3	8	4	9	5	10
1	9	5	10	1	6	2	7	3	8	4
2	3	8	4	9	5	10	1	6	2	7
3	6	2	7	3	8	4	9	5	10	1
4	5	10	1	6	2	7	3	8	4	9
5	8	4	9	5	10	1	6	2	7	3
6	2	7	3	8	4	9	5	10	1	6
7	10	1	6	2	7	3	8	4	9	5
8	4	9	5	10	1	6	2	7	3	8
9	7	3	8	4	9	5	10	1	6	2

Appendix B
FDOS Global References

The following globals reside in low memory (>80 to >100) and can be used by application programs to monitor the state of FDOS as well as to find out where certain routines or data areas of FDOS reside.

NAME	LOCATION	USE
RDRECT	>80	This location points to read the directory routine.
WDRECT	>82	This location points to write the directory routine.
DIR	>84	This location points to the beginning of the memory resident directory.
BUFFER	>86	This location points to the read/write buffer.
DEVNAM	>88	This location points to the device name table. This table is built during initialization of the operating system and consists of the RADIX-50 representation of the device name of each linked-in module. A dummy entry is generated as the last item in the device table. This entry can be used by any program to generate a temporary device. When control is given back to FDOS via a SVC EXIT or a CTRL/P, this device will be deleted from the table.
DEVNUM	>8A	This location points to the device number table. This table is built during initialization of the operating system and consists of a byte for each device. The four lower bits of the byte represent the XOP number by which the device is accessed. Bit 4 of this byte is set to indicate that the device is non-file structured. The device numbers appear in the same order as their corresponding device names.
CURDIR	>8C	This location points to the location holding the key for the current directory. This key consists of the XOP instruction by which the driver of the device associated with the current directory can be accessed. The argument of this XOP instruction is always R0. For example, the XOP for driver 3 is XOP R0,3.

NAME	LOCATION	USE
DEVSIZ	>8E	This location points to the device size table. This table is built during initialization of the operating system and consists of a word indicating how many blocks are available on the dveice. The size numbers appear in the same order as their corresponding device names.
AREA	>A4	This location points to the parameter passing area. This location should be read only.
DATE	>A6	This location holds the current date, read only. See Appendix A for the format.
TIME1	>A8	This location containsn the upper portion of current time, read only.
TIME2	>AA	This location contains the lower portion of current time, read only. Time is expressed as a two word integer that indicates the time elapsed since midnight in 10 msec increments.
CL	>E6	This location points to the first character of the command line. The command line is the user or command file entry following the prompt CONMON.
AA	>EA	This location points to the argument area. This area is used to pass an argument into a command file.
CW	>EC	This location points to the command file processor workspace.
CB	>EE	This location points to the buffer that contains the command file.
AP	>F0	This location points to the first byte of the argument for the command file. If this location is zero, then no argument was given.
KEY	>AC	This location contains the last character (as a word) entered from the console terminal with the parity bit set (overlay). This word may be cleared after reading.
VERIFY	>F2	This location is a flag. This flag indicates to certain device drivers that a read after write has to be performed in order to insure the integrity of written data.
RETRY	>F4	This location displays the number of retries a deviec handler needed to read a block of data. This number is reset every time the handler is entered again.

NAME	LOCATION	USE
CF	>F6	This location is the command file active flag. See the command file interpreter module (Section 5) for a detailed explanation of this flag.
CC	>F8	This location is the CTRL/C flag. This flag is set to a value other than zero every time a CTRL/C or CTRL/P is sent by the console terminal. The user can reset this flag.
OPWORK	>FA	This location points to the first memory location occupied by FDOS. This location can be used to show the top of available memory.